НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 004.94(075.8)

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Аналіз та моделювання дискретно-подійного процесу на основі журналу подій»*

Виконав: студент *4* курсу, групи *ІС-51*

Рябцев Сергій Вячеславович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *проф., д.т.н., доц. Стеценко І.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *старший викладач Москаленко Н.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Рябцеву Сергію Вячеславовичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Аналіз та моделювання дискретно-подійного процесу на основі журналу подій*»

керівник проекту *Стеценко Інна В'ячеславівна, д.т.н, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема структурна діяльності

3. Схема структурна послідовності

4. Схема структурна класів програмного забезпечення

5. Креслення вигляду екранних форм

6. Креслення вигляду звітних форм

7. Рішення з математичного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«15» лютого 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*20.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*29.02.2019*	
3.	*Постановка та формалізація задачі*	*05.03.2019*	
4.	*Розробка інформаційного забезпечення*	*15.03.2019*	
5.	*Алгоритмізація задачі*	*12.04.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*18.04.2019*	
7.	*Розробка програмного забезпечення*	*20.05.2019*	
8.	*Налагодження програми*	*27.05.2019*	
9.	*Виконання графічних документів*	*15.05.2019*	
10.	*Оформлення пояснювальної записки*	*27.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ Рябцев С.В.
(підпис)

Керівник проекту _____ Стеценко І.В.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5123.1181-с.ПЗ	Пояснювальна записка	109	
2	А4	ДП ІС-5123.1181-с.ТЗ	Технічне завдання	15	
3	А4	ДП ІС-5123.1181-с.ПМВ	Програма та методика випробувань	18	
			Матеріали графічні		
4	А3	ДП ІС-5123.1181-с.ССВ	Схема структурна варіантів використань	1	
5	А3	ДП ІС-5123.1181-с.ССД	Схема структурна діяльності	1	
6	А3	ДП ІС-5123.1181-с.ССП	Схема структурна послідовності	1	
7	А3	ДП ІС-5123.1181-с.ССК	Схема структурна класів програмного забезпечення	1	
8	А3	ДП ІС-5123.1181-с.КЕ	Креслення вигляду екранних форм	1	
9	А3	ДП ІС-5123.1181-с.КЗ	Креслення вигляду звітних форм	1	
10	А3		Рішення з математичного забезпечення	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5123.1181-с.ВДП		
Розробн.		Рябцев С.В.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Стеценко І.В.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Н/контр.		Москаленко Н.В.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Аналіз та моделювання дискретно-подійного процесу на основі журналу подій

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 25 рисунків, 21 таблицю, 1 додаток, 22 джерел.

Дипломний проект присвячений розробці програмного продукту, призначеного для виявлення, аналізу та візуалізації моделі дискретно-подійного процесу на основі журналу подій.

Мета роботи – розробка, дослідження та використання реалізації евристичного алгоритму виявлення процесу (англ. Heuristic Miner) для аналізу та моделювання дискретно-подійного процесу.

У розділі інформаційного забезпечення було визначено структуру вхідних та вихідних даних і структуру масивів інформації.

Розділ математичного забезпечення присвячений формулюванню математичної постановки задачі, пошуку існуючих рішень та обґрунтуванню вибору евристичного алгоритму процесної аналітики для рішення даних задачі.

У розділі програмного забезпечення було описано та обґрунтовано засоби розробки та архітектурні рішення.

У технологічному розділі наведено керівництво користувача та методику випробувань розробленого програмного продукту.

БІЗНЕС-ПРОЦЕС, ДИСКРЕТНО-ПОДІЙНИЙ ПРОЦЕС, ПРОЦЕСНА АНАЛІТИКА, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, МЕРЕЖА ПЕТРІ, ЖУРНАЛ ПОДІЙ

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of five sections, contains 25 pictures, 21 tables, 1 application, 22 references.

The diploma project is devoted to the development of a software product intended for discovery, analysis and visualization of the discrete-event process model based on an event log.

The aim of the work is to develop, research and use the implementation of the Heuristics Miner algorithm for analysis and modeling of discrete event process.

In the information support section, the structure of input and output data and the structure of information arrays were determined.

The section of mathematical support is devoted to the formulation of mathematical statement of the problem, search for existing solutions and justification of the choice of Heuristics Miner algorithm of Process Mining to solve these problems.

In the software section, the means of development and architectural solutions were described and justified.

The technological section contains the user's manual and test methods of the developed software product.

BUSINESS PROCESS, DISCRETE EVENT PROCESS, PROCESS MINING, SIMULATION MODELING, PETRI NET, EVENT LOG

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Алгоритм Альфа (англ. Alpha Miner) – один з перших алгоритмів процесної аналітики, що виявляє процес (у вигляді мереж Петрі) з журналів подій.

Алгоритм нечіткого виявлення (англ. Fuzzy Miner) – алгоритм процесної аналітики, що підлаштовується і дозволяє виявляти різні вірно спрощені уявлення конкретного процесу. Такий алгоритм підходить для виявлення менш структурованих процесів, які демонструють велику кількість неструктурованих і конфліктуючих поведінок.

Генетичний алгоритм (англ. Genetic Miner) – алгоритм процесної аналітики, що є адаптивним методом пошуку, що намагається імітувати процес еволюції. Індивідуум є можливим процесом, а придатність - це функція, яка оцінює, наскільки добре він здатен відтворювати поведінку в журналі.

Евристичний алгоритм (англ. Heuristics Miner) – алгоритм процесної аналітики, що може працювати з шумом і може використовуватися для вираження основної поведінки (тобто не всіх деталей і виключень), зареєстрованих у журналі подій.

Журнал подій (англ. Event Log) – набір даних, що містить інформацію службового чи статистичного характеру про події в системі.

Процесна аналітика (англ. Process Mining) – загальна назва ряду методів і підходів, призначених для аналізу та удосконалення процесів в інформаційних системах або бізнес-процесів на підставі вивчення системних даних про виконані операції в системі. Основна ідея полягає в отриманні знань про структуру і поведінку процесу з журналів подій, що створюються інформаційними системами під час функціонування.

ВСТУП

У даній дипломній роботі розглядається задача виявлення моделі дискретно-подійного процесу на основі журналу подій (англ. Event Log) за допомогою евристичного алгоритму (англ. Heuristics Miner) процесної аналітики (англ. Process Mining). Іншими словами, отримання схематичного зображення процесу та зв’язків між його елементами у вигляді мережі Петрі на основі вхідних даних – журналу подій (логів) [1].

Як відомо, евристичні алгоритми – це алгоритми рішення задач, правильність яких для всіх можливих випадків не доведена, але про які відомо, що вони дають відносно точне рішення в більшості випадків. Евристика – це не повністю математично обґрунтований, але при цьому практично корисний алгоритм.

Важливо розуміти, що евристика, на відміну від коректного алгоритму розв'язання задачі, володіє наступними особливостями:

- вона не гарантує знаходження кращого рішення;
- вона не гарантує знаходження рішення, навіть якщо воно явно існує;
- вона може дати невірне рішення в деяких випадках.

Процесна аналітика – загальна назва ряду методів і підходів, призначених для аналізу та удосконалення процесів в інформаційних системах або бізнес-процесів на підставі вивчення системних даних про виконані операції в системі. Основна ідея полягає в отриманні знань про структуру і поведінку процесу з журналів подій, що створюються інформаційними системами під час функціонування [2].

Запропонований програмний продукт призначений для виявлення, аналізу та візуалізації моделі дискретно-подійного процесу на основі журналу подій.

Мета роботи – розробка, дослідження та використання реалізації евристичного алгоритму виявлення процесу (англ. Heuristics Miner) для аналізу та моделювання дискретно-подійного процесу.

Для реалізації поставленої мети роботи необхідно розв’язати наступні задачі:

- розробка та реалізація евристичного алгоритму виявлення моделі процесів;
- розробка та реалізація алгоритму перетворення виявленого процесу у мережу Петрі;
- створення засобів проведення аналізу моделі дискретно-подійного процесу на основі журналу подій;
- створення засобів графічної візуалізації виявленої моделі та результатів її аналізу.

Практичне значення одержаних результатів. Розроблено алгоритм виявлення, аналізу та перетворення у мережу Петрі дискретно-подійного процесу на основі даних журналу подій.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Бізнес-процеси керують та підтримують більшість функцій та послуг на підприємствах та адміністративних органах сучасного світу. Для опису таких процесів, моделювання їх як діаграм та моделей виявилося корисним і інтуїтивно зрозумілим інструментом. Попри те, що моделювання є широко застосоване в процесі проектування, його досить нечасто використовують з метою моніторингу або документації. Однак, особливо для моніторингу, процесні моделі є цінними артефактами, тому що вони дозволяють передавати складні знання в інтуїтивно зрозумілому і компактному вигляді. Процес отримання таких моделей – це ряд досліджень, які намагаються витягти такі абстрактні, компактні зображення процесів з їхніх журналів (журналів подій, англ. event logs), тобто історії їх виконання. Застосовувані до чітко спроектованих, добре структурованих і вдало запроваджених процесів, ці методи здатні забезпечити вражаючий набір інформації, проте їхня мета дещо обмежена перевіркою відповідності виконуваним вимогам. Однак більшість процесів у реальному житті не були цілеспрямовано розроблені та оптимізовані, розвивалися з часом або навіть не були чітко визначені. У таких ситуаціях застосування виявлення процесу є набагато кориснішим, оскільки в цьому випадку воно не обмежується повторним виявленням того, що вже відомо, а надає можливість розкривати раніше приховані знання.

Процесна аналітика (англ. Process mining) – є сферою досліджень, яка пов'язана з апостеріорним аналізом бізнес-процесів на основі журналів подій їх виконання. Вона має на меті отримати з цих журналів загальну інформацію високого рівня про деякі аспекти процесу. Наприклад, техніка виявлення процесу може генерувати модель процесу з журналу подій, що описує спостережуваний потік управління процесом. На відміну від інших методів аналізу, методи виявлення процесу можуть дати точну та фактичну

інформацію про процес, а не покладатися на ідеалізовану модель реальності. Це робить його потужним інструментом для аналізу процесів, тобто, для відкриття точної картини поточної ситуації, що дозволить зробити відповідні висновки і поліпшити характеристики процесу [1].

Більшість бізнес-процесів сьогодні підтримуються інформаційними системами, і щодня виконується величезна кількість процесних подій. Навіть для одного окремого процесу в одній організації це призводить до великих обсягів даних журналу подій, які неможливо проаналізувати без комп'ютерної підтримки. Проте бізнес-процеси не є єдиним джерелом даних журналу подій. Все частіше, складні пристрої та системи (наприклад, рентгенівські прилади або системи управління залізницею) покладаються на внутрішні процеси для своєї роботи, і записують докладні журнали подій про їх використання. Більше того, люди беруть участь у багатьох великих веб-системах щодня. Участь у соціальних веб-спільнотах, подання пошукових запитів і спілкування за допомогою електронної пошти можна розглядати як частину великих процесів, які ретельно записуються в центральні журнали подій (наприклад, журнали доступу до веб-серверів або журнали серверів додатків, веб-сайти, що надають ці послуги). Явище, коли багато наших повсякденних дій неявно записується в базах даних і журналах подій, і кількість створених даних, часто називають вибухом даних. З кожним роком збільшується кількість процесів, виконання яких записується в журналах подій.

Таким чином, існує реальна і нагальна потреба в таких методах, як виявлення процесів, які можуть допомогти в аналізі зв’язаної з процесами інформації, записаної в журналах подій. Для фактичного виявлення моделі процесу, заснованої на таких журналах подій, поле процесної аналітики забезпечує велику кількість методів, які можна застосовувати. Більшість алгоритмів виявлення процесу можуть аналізувати великі обсяги подій за відносно короткий час, так що виявлення процесу може виконуватися в

інтерактивному режимі і таким чином може стати частиною щоденної процедури для діагностики процесу.

Метод виявлення процесу також розглядає інші перспективи аналізованого процесу, наприклад, перспективу даних або організаційну перспективу. Більшість журналів подій містять джерела для кожної події, тобто особу (або ресурс), яка виконала кожне завдання. Враховуючи цю інформацію, для процесу виявлення можна створити соціальну мережу цих учасників, описуючи, хто співпрацює з ким і хто має центральну роль в організації. Хоча ці приклади стосуються виявлення оригінальних моделей процесу, що генеруються тільки з журналу подій, методи виявлення можуть також займатися оцінкою відповідності та розширеним аналізом.

Під оцінкою відповідності мова йде про кількісні оцінки, наскільки тісно модель даного процесу відповідає дійсності, як це спостерігається в журналах подій. Алгоритми розширеного аналізу можуть бути використані для доповнення даної моделі процесу додатковою інформацією, що міститься в зведених таблицях. Одним з прикладів є аналіз продуктивності процесу та проекціювання цієї інформації на модель управління процесом. Це може бути дуже корисним інструментом для виявлення проблемних частин (наприклад, вузьких місць) в потоці управління або в організації.

У сфері процесної аналітики було створено велику кількість методів виявлення, що стосуються багатьох перспектив процесу з точки зору власне виявлення, оцінки відповідності та розширеного аналізу [1-3]. Попри це все, потенціал аналізу моделі процесів не обмежується лише сферою процесної аналітики. З’являється багато нових можливостей і напрямів для аналізу бізнес-процесів з переходом у сферу імітаційного моделювання.

Імітаційні моделі мають на меті повторити роботу та логіку реальної системи, використовуючи статистичні описи задіяних заходів. Імітаційна модель має «сутності» (наприклад, машини, матеріали, люди тощо) і «діяльність» (наприклад, обробка, транспортування тощо). Вона також має

опис логіки, що регулює кожну діяльність. Після початку діяльності розраховується час до завершення, часто використовуючи вибірку зі статистичного розподілу.

Зрозуміло, що імітаційні моделі можуть повторювати складну систему виробництва. Вони можуть бути використані для позначення рівня спільних ресурсів, необхідних операції (наприклад, навантажувачів або операторів), швидкості ліній, розмірів суден або резервуарів для зберігання тощо.

Розглянемо переваги використання імітаційних моделей. По-перше, імітаційне моделювання надає безпечний спосіб перевірити і дослідити різні сценарії типу «що-якщо». Далі, віртуальні експерименти з імітаційними моделями менш дорогі і займають менше часу, ніж експерименти з реальними активами. Імітаційні моделі можуть бути анімовані в 2D/3D, що дозволяє більш зручно перевіряти, передавати та розуміти концепції та ідеї. На відміну від аналітики, що використовує електронні таблиці, імітаційне моделювання дозволяє спостерігати за поведінкою системи в часі, теоретично на будь-якому рівні деталізації. Недетермінованість часових затримок та результатів роботи може бути досліджена в імітаційних моделях з використанням генераторів випадкових чисел, що надає можливість кількісно оцінювати ризики та обирати найбільш надійні рішення.

Імітаційна модель може зафіксувати набагато більше деталей, ніж аналітична модель, забезпечуючи підвищену точність і більш точне прогнозування. До недоліків таких моделей, як правило, відносять складність і, як наслідок, вартість та тривалість розробки такої моделі.

Проаналізувавши усе це, можна дійти висновку, що потенціал процесної аналітики стає в рази потужнішим, якщо з’являється змога використати результати роботи алгоритмів виявлення процесів для побудови імітаційної моделі бізнес-процесу.

1.1.1 Опис процесу діяльності

Опис процесу діяльності представлено у графічному матеріалі у вигляді діаграми діяльності. Уся діяльність розподілена між двома ролями: користувача та програмного продукту.

Як можна зазначити з діаграми діяльності, користувач завантажує вхідні дані, задає параметри виявлення моделі процесу. Після цього програмний продукт виявляє дискретно-подійний процес та візуалізує його у вигляді мережі Петрі. Користувач переглядає дану візуалізацію. Потім програмний продукт аналізує виявлений дискретно-подійний процес та знову ж таки візуалізує його для користувача. Користувач, в свою чергу, переглядає звіт з аналізу процесу та завершує роботу з програмним продуктом.

1.1.2 Опис функціональної моделі

Специфікацію функціональної поведінки системи представлена у вигляді діаграми варіантів використання, що наведена у графічному матеріалі.

Усі варіанти використання системи задіяні одним актором – користувачем. Користувач має можливість завантажувати вхідні дані; задавати параметри виявлення процесу, що в свою чергу включать в себе задання стовпчиків діяльності, випадків, часової мітки початку, часової мітки завершення та додаткових параметрів (порогових значень кількості прямих наслідувань, міри зв’язності, значення розміру вікна для встановлення розгалужень та з’єднань); виявляти дискретно-подійний процес за заданими вхідними даними та вказаними параметрами; аналізувати виявлену модель процесу; переглядати візуалізовану мережу Петрі та візуалізовані результати проведеного аналізу побудованої моделі дискретно-подійного процесу.

Відповідно до зазначених варіантів використання було складено функціональні вимоги із зазначенням їх пріоритету. Результат наведено в таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги

Варіант використання	Функціональна вимога	Пріоритет
Завантаження вхідних даних	1. Система повинна надавати можливість завантажувати файл вхідних даних	Високий
	1.1. Система повинна надавати можливість завантажувати файли з розширенням .csv (англ. Comma-Separated Values)	Високий
	1.1.1. Система повинна встановлювати фільтр на відображення лише файлів з розширенням .csv підчас вибору файлу для завантаження	Низький
	1.2. Система повинна візуалізувати завантажені вхідні дані у вигляді таблиці	Середній
	1.2.1. Система повинна надавати можливість змінювати порядок стовпців у візуалізованій таблиці вхідних даних	Низький
	1.2.2. Система повинна надавати можливість відсортувати у висхідному чи низхідному порядку будь-який стовпець візуалізованої таблиці вхідних даних	Низький
Задання параметрів виявлення процесу	2. Система повинна надавати можливість задання параметрів виявлення дискретно-подійного процесу	Середній
Задання стовпчику випадків	2.1. Система повинна надавати можливість задання стовпчику випадків	Середній

Продовження таблиці 1.1

	2.1.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику діяльності	2.2. Система повинна надавати можливість задання стовпчику діяльності	Середній
	2.2.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику часової мітки початку	2.3. Система повинна надавати можливість задання стовпчику часової мітки початку	Середній
	2.3.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику часової мітки завершення	2.4. Система повинна надавати можливість задання стовпчику часової мітки завершення	Середній
	2.4.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький

Продовження таблиці 1.1

Задання додаткових параметрів	2.5. Система повинна надавати можливість задання додаткових параметрів	Середній
	2.5.1. Система повинна надавати можливість задати числові параметри: порогове значення кількості прямих наслідувань, порогове значення міри зв'язності, значення розміру вікна для встановлення розгалужень та з'єднань	Середній
	2.5.2. Система повинна надавати можливість задання додаткових параметрів за допомогою «повзунків» з обмеженим діапазоном. 2.5.3. Система повинна встановлювати діапазон допустимих значень додаткових параметрів	Низький
	2.5.4. Система повинна не надавати можливість задання таких значень додаткових параметрів, що знаходять поза діапазону допустимих значень	Низький
Виявлення дискретно-подійного процесу	3. Система повинна за вхідними даними та заданими параметрами виявляти дискретно-подійний процес у вигляді мережі Петрі	Високий
	3.1. Система повинна за допомогою евристичного алгоритму виявлення моделі процесу виявляти граф залежності процесу	Високий
	3.2. Система повинна перетворювати граф залежності процесу у мережу Петрі	Високий
	3.3. Система повинна спрощувати отриману мережу Петрі	Середній

Продовження таблиці 1.1

Аналіз дискретно-подійного процесу	4. Система повинна за вхідними даними та заданими параметрами аналізувати дискретно-подійний процес	Високий
	4.1. Система повинна розраховувати кількість спрацьовувань кожної з виявлених діяльностей у чисельному та відсотковому вимірах	Високий
	4.2. Система повинна розраховувати кількість завершених переходів між будь-якими двома з виявлених діяльностей у чисельному та відсотковому вимірах	Високий
	4.3. Система повинна розраховувати мінімальне, середнє, та максимальне значення тривалості виконання кожної з виявлених діяльностей	Високий
	4.4. Система повинна розраховувати мінімальне, середнє, та максимальне значення тривалості затримок між виконанням будь-яких двох з виявлених діяльностей	Високий
Перегляд візуалізованої мережі Петрі	5. Система повинна надавати можливість переглянути виявлений дискретно-подійний процес у вигляді мережі Петрі	Високий
	5.1. Система повинна візуалізовувати отриману мережу Петрі у вигляді набору сполучених між собою «місць» та «переходів», дотримуючись усіх нотацій мереж Петрі	Високий
	5.2. Система повинна надавати можливість пересувати кожен з елементів візуалізованої мережі Петрі	Низький
	5.3. Система повинна надавати можливість редагувати кожен елемент візуалізованої мережі Петрі	Низький

Продовження таблиці 1.1

	5.4. Система повинна надавати можливість видаляти елементи візуалізованої мережі Петрі	Низький
	5.5. Система повинна надавати можливість додавати нові елементи до візуалізованої мережі Петрі	Низький
Перегляд візуалізованого результату аналізу	6. Система повинна надавати можливість переглянути візуалізовані результати аналізу дискретно-подійного процесу	Високий
	6.1. Система повинна візуалізувати результати аналізу в окремому від візуалізації мережі Петрі вікні	Середній
	6.2. Система повинна згрупувати отримані результати по тематичними групам: «Частота діяльності», «Частота переходів», «Тривалість діяльності» та «Тривалість затримок»	Середній
	6.3. Система повинна відображати як повну назву діяльності, так і її буквене позначення, введене системою	Низький
	6.4. Система повинна відображати чисельні значення оцінки часу в секундах	Низький
	6.5. Система повинна надавати можливість пересувати довільним чином стовпці кожної з таблиць	Низький
	6.6. Система повинна надавати можливість сортувати у висхідному та низхідному порядках вміст кожного стовпця кожної таблиці	Середній

1.2 Огляд наявних аналогів

На даний момент часу у світі існує декілька десятків програмних продуктів, що реалізують алгоритми процесної аналітики, виявляють та

аналізують моделі дискретно-подійних процесів. До провідних представників таких продуктів можна віднести ProM, Disco, Celonis та myInvenio.

ProM – це програмне забезпечення з відкритим вихідним кодом, написаним на Java. Тобто річ йде про кросплатформове програмне забезпечення. Віл ван дер Аалст та його дослідницька група розробляють ProM в Ейндховенському технологічному університеті (Нідерланди). Це потужний інструмент з багатьма алгоритмами і функціями. Існує велика кількість плагінів для різних алгоритмів виявлення, аналізу, перетворення та експорту модулів. Програмний інструмент спрямований переважно на академічну та дослідницьку групу [4].

Disco є комерційним інструментом, розробленим компанією Fluxicon, але він має академічну ліцензію з повною підтримкою. Disco працює на Windows або MacOS X. Підтримує широкий спектр форматів імпорту журналів подій. До ключових особливостей можна віднести автоматизоване виявлення процесів, анімацію процесних карт, фільтрацію журналу подій з різними параметрами, управління проектами та детальну статистику [5].

Celonis – ще один комерційний продукт, що також має академічну ліцензію. Він працює на Windows або Mac OSX. Celonis пропонує техніку виявлення моделі процесів в реальному часі, тобто підчас функціонування досліджуваної інформаційної системи. Серед ключових особливостей цього інструменту - автоматизована інтеграція вихідних даних, спостереження в реальному часі всіх бізнес-операцій, виконання аналізу процесів, різноманітні механізми фільтрації та звітність процесу [6].

myInvenio – комерційний продукт з можливістю отримання академічної ліцензії. Однак, це веб-застосунок. Таким чином, він може бути доступний з будь-якого пристрою (мобільного, планшетного, комп'ютерного), який має браузер і активне підключення до Інтернету [7].

Порівняльний аналіз даних програмних продуктів сформульовано та узагальнено у таблиці 1.2 [8].

Таблиця 1.2 – Порівняльний аналіз програмних продуктів

Особливості	Програмний продукт			
	ProM	Disco	Celonis	myInvenio
Ліцензія	Відкритий код	Комерційна	Комерційна	Комерційна
Платформа	Настільні ПК	Настільні ПК	Настільні ПК, Веб	Веб
Фільтрування вхідних даних	Так	Так	Так	Так
Дружелюбний користувацький інтерфейс	Ні	Так	Так	Так
Виявлення моделі процесу	Так	Так	Так	Так
Реалізація евристичного алгоритму виявлення процесів	Так	Ні	Ні	Ні
Вихідні дані	BPMN, Fuzzy model, Мережа Петрі, Heuristics model, Transition system	Fuzzy model	Fuzzy model, діаграми	BPMN, SVG, CSV
Аналіз отриманої моделі	Так	Так	Так	Так
Візуалізація моделі	Так	Так	Так	Так
Середовище для імітаційного моделювання	Ні	Ні	Ні	Ні
Редагування виявленої моделі	Ні	Ні	Ні	Ні

Можна сказати, що ProM є незамінним інструментом для виявлення моделі процесів. Тому, що він з відкритим кодом, постійно розширюється і має всі функції, необхідні для виявлення процесів.

Хоча ProM є найкращим серед розглянутих рішень, він має проблему з інтерфейсом користувача. Використовувати його не так просто, як будь-які інші інструменти. Це пов’язано з тим, що даний програмний продукт розроблюється науковими дослідниками для наукових дослідників. Тобто річ йде скоріше про професійний інструмент, ніж про продукт широкого користувацького вжитку. Це і є основною причиною того, що в останні часи з’являється все більше комерційних, вузько спеціалізованих інструментів, розрахованих на звичайних користувачів [9].

Крім цього, ProM єдиний з розглянутих програмних продуктів, що може виявляти моделі процесів у вигляді мереж Петрі. Але й в ньому функціонал для роботи з виявленою мережею дуже обмежений. Немає будь-якої можливості редагування мереж Петрі. Це стає особливо важливим, коли йдеться про імітаційне моделювання. В розглянутих програмних продуктах не має і натяку стосовно цього напрямку використання виявлених моделей процесів.

Отже, звідси і витікають ідеї доцільності створеного в рамках дипломного проекту програмного продукту.

1.3 Постановка задачі

1.3.1 Призначення розробки

Запропонований програмний продукт призначений для виявлення, аналізу та візуалізації моделі дискретно-подійного процесу на основі журналу подій.

1.3.2 Цілі та задачі розробки

Мета роботи – розробка, дослідження та використання реалізації евристичного алгоритму виявлення процесу (англ. Heuristics Miner) для аналізу та моделювання дискретно-подійного процесу.

Для реалізації поставленої мети роботи необхідно розв’язати наступні задачі:

- розробка та реалізація евристичного алгоритму виявлення моделі процесів;
- розробка та реалізація алгоритму перетворення виявленого процесу у мережу Петрі;
- створення засобів проведення аналізу моделі дискретно-подійного процесу на основі журналу подій;
- створення засобів графічної візуалізації виявленої моделі та результатів її аналізу.

Висновок до розділу

Процесна аналітика є методом виявлення моделей процесу з журналів подій. Метод виявлення процесу, який спрямований на вилучення пов'язаної з процесом інформації з журналів подій, став важливим інструментом для сучасних організацій, яким необхідно керувати нетривіальними операційними процесами. Можливість автоматизованої розробки імітаційної моделі за результатами виявлення процесу та відсутність серед аналогів програмних продуктів з подібним функціоналом формулюють передумови до створення власного програмного продукту, що реалізує вищезгаданий функціонал.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідними даними є журнал подій, представлений у вигляді таблиці. Приклад журналу подій представлено на рисунку 2.1.

case id	activity	start time	finish time	cost	device id
1	register order	12.01.2019 12:04:01	12.01.2019 12:04:23	3	354353
2	register order	12.01.2019 12:04:12	12.01.2019 12:04:27	3	471356
1	check stock	12.01.2019 12:04:25	12.01.2019 12:04:29	12	354353
3	register order	12.01.2019 12:04:47	12.01.2019 12:04:59	3	150087
1	ship order	12.01.2019 12:04:56	12.01.2019 12:05:04	15	354353
3	check stock	12.01.2019 12:05:06	12.01.2019 12:05:15	10	150087
1	handle payment	12.01.2019 12:05:07	12.01.2019 12:05:18	1	354353
1	exit	12.01.2019 12:05:21	12.01.2019 12:05:29	0	354353
2	check stock	12.01.2019 12:05:54	12.01.2019 12:06:06	10	471356
2	cancel order	12.01.2019 12:06:10	12.01.2019 12:06:21	5	471356
2	exit	12.01.2019 12:06:11	12.01.2019 12:06:16	0	471356
3	exit	12.01.2019 12:07:01	12.01.2019 12:07:05	0	150087

Рисунок 2.1 – Приклад журналу подій

Журнал подій має містити щонайменше 4 стовпці: стовпець ідентифікатора випадку (case id), стовпець діяльності (activity), стовпці мітки часу початку діяльності (start time) та її завершення (finish time). Окрім зазначених стовпців, журнал подій може містити й іншу пов’язану з процесом інформацію (наприклад, стовпці cost, device id).

Порядок стовпців та їх назви можуть бути довільними. Елементами стовпця ідентифікатора випадку мають бути цілі невід’ємні числа. Елементами стовпця діяльності мають бути довільні рядкові значення. Елементами стовпців міток часу початку та завершення мають бути рядкові значення із заданням дати і часу.

2.2 Вихідні дані

До вихідних даних належать візуалізований виявлений дискретно-подійний процес у вигляді мережі Петрі та результати аналізу даного процесу.

Мережа Петрі являє сполучені між собою елементи: «позиції» та «переходи» [10]. Приклад мережі Петрі зображено на рисунку 2.2.

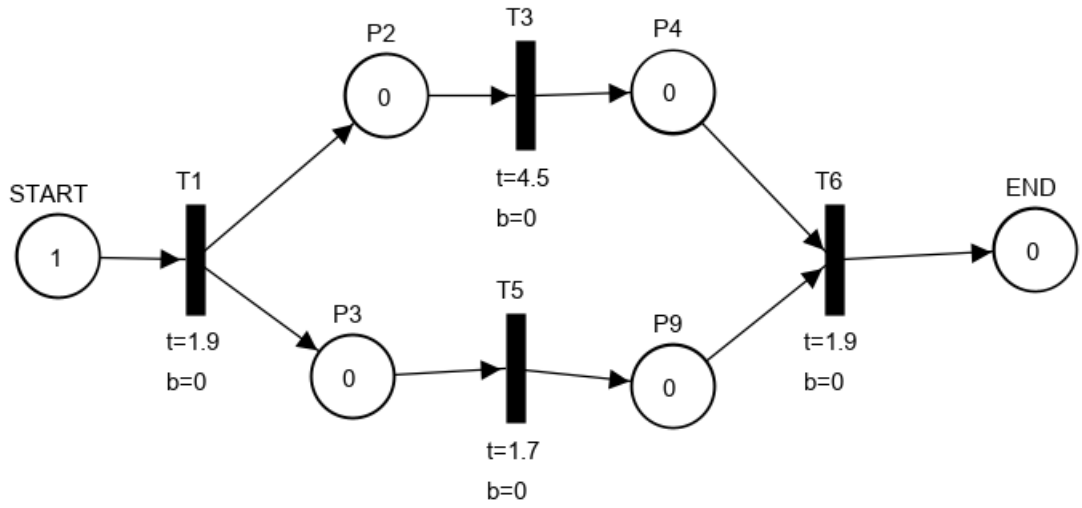


Рисунок 2.2 – Приклад мережі Петрі

Звіт з аналізу дискретно-подійного процесу складається з чотирьох таблиць: «Частота діяльності», «Частота переходу», «Тривалість діяльності» та «Тривалість затримки». Структура наведених таблиць представлена нижче в таблицях 2.1-2.4.

Таблиця 2.1 – Частота діяльності

Повна назва	Буквене позначення	Частота	Частота (%)

Таблиця 2.2 – Частота переходу

Звідки (повна назва)	Звідки (буквене позначення)	Куди (повна назва)	Куди (буквене позначення)	Частота	Частота (%)

Таблиця 2.3 – Тривалість діяльності

Повна назва	Буквене позначення	Мінімальне (мс)	Середнє (мс)	Медіана (мс)	Максимальне (мс)

Таблиця 2.4 – Тривалість затримки

Звідки (повна назва)	Звідки (буквене позначення)	Куди (повна назва)	Куди (буквене позначення)	Мінімальне (мс)	Середнє (мс)	Медіана (мс)	Максимальне (мс)

2.3 Структура масивів інформації

Головним та єдиним масивом інформації програмного забезпечення є текстове подання журналу подій. Приклад вхідного файлу в табличній візуалізації зображено на рисунку 2.1, в текстовому поданні – на рисунку 2.3 нижче:

```
case id;activity;start time;finish time;cost;device id
1;register order;12.01.2019 12:04:01;12.01.2019 12:04:23;3;354353
2;register order;12.01.2019 12:04:12;12.01.2019 12:04:27;3;471356
1;check stock;12.01.2019 12:04:25;12.01.2019 12:04:29;12;354353
3;register order;12.01.2019 12:04:47;12.01.2019 12:04:59;3;150087
1;ship order;12.01.2019 12:04:56;12.01.2019 12:05:04;15;354353
3;check stock;12.01.2019 12:05:06;12.01.2019 12:05:15;10;150087
1;exit;12.01.2019 12:05:21;12.01.2019 12:05:29;0;354353
2;check stock;12.01.2019 12:05:54;12.01.2019 12:06:06;10;471356
2;cancel order;12.01.2019 12:06:10;12.01.2019 12:06:21;5;471356
2;exit;12.01.2019 12:06:11;12.01.2019 12:06:16;0;471356
3;exit;12.01.2019 12:07:01;12.01.2019 12:07:05;0;150087
```

Рисунок 2.3 – Приклад текстового подання журналу подій

Кожен рядок вхідного файлу відповідає рядку таблиці журналу подій. Стовпці таблиці журналу подій в текстовому подані розділені символом «;».

Таким чином, вхідний файл має містити щонайменше 4 стовпці: стовпець ідентифікатора випадку (case id), стовпець діяльності (activity), стовпці мітки часу початку діяльності (start time) та її завершення (finish time). Окрім зазначених стовпців, журнал подій може містити й іншу пов’язану з

процесом інформацію (наприклад, стовпці cost, device id), причому порядок стовпців та їх назви можуть бути довільними.

На вміст таблиці вхідних даних задані наступні обмеження. Стовпець ідентифікатора випадку має містити лише цілі невід'ємні числа з проміжку від 0 до 100000 включно. Стовпець діяльності має містити лише рядкові значення, довжиною символів від 1 до 200 включно. Стовпці міток часу початку та завершення мають містити лише рядкові значення із заданням дати і часу у форматі "dd.MM.yyyy HH:mm:ss", де:

- dd – номер дня, ціле двоцифрове число від 1 до 31 включно;
- MM – номер місяця, ціле двоцифрове число від 1 до 31 включно;
- yyyy – рік, ціле чотирьох цифрове число від 1000 до 9999 включно;
- HH – кількість годин, ціле двоцифрове число від 0 до 23 включно;
- mm – кількість хвилин, ціле двоцифрове число від 0 до 59 включно;
- ss – кількість секунд, ціле двоцифрове число від 0 до 59 включно.

Наприклад, "12.01.2019 12:06:11".

Висновок до розділу

До вхідних даних відносять журнал подій зі стовпцями, що містять інформацію про ідентифікатор випадку, назву діяльності та часові мітки початку та кінця діяльності.

До вихідних даних належить візуалізована у вигляді графу мережа Петрі, побудована на основі виявленого дискретно-подійного процесу, та звіт з аналізу виявленого процесу у вигляді чотирьох таблиць із зазначенням частоти та тривалості окремих діяльностей та переходів.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Інформаційна система в ході свого функціонування веде журнал подій, що відбулись. Записи ведуться в табличному вигляді із зазначенням такої інформації:

- ідентифікатор випадку (case id) – події, що відбулися в рамках одного випадку (варіанту використання / одним користувачем / за одну сесію взаємодії з інформаційною системою) мають один унікальний ідентифікатор – ціле невід'ємне число;
- назва / ідентифікатор діяльності (activity) – рядкове значення, яке певним чином описую діяльність, що відбулась;
- мітка часу початку діяльності (start timestamp) – дата та час початку діяльності. Рядкове значення формату "dd.MM.yyyy HH.mm.ss";
- мітка часу завершення діяльності (finish timestamp) – дата та час завершення діяльності. Рядкове значення формату "dd.MM.yyyy HH.mm.ss".

На виявлення дискретно-подійного процесу накладається ряд обмежень, а саме:

- мінімальне значення прямих слідувань – кількість разів, коли одна діяльність відбувалась відразу після іншої;
- мінімальне значення міри зв'язності – дійсне число в діапазоні від -1 до 1, зо вказує на однозначність переходу з однієї діяльності на іншу.
- розмір діапазону для дослідження структури розгалужень та сполучень між діяльностями – встановлює кількість наступних і попередніх діяльностей відносно досліджуваної, що будуть розглянуті для встановлення типу розгалуження чи сполучення.

Маючи довільну додатну кількість записів у журналі подій потрібно виявити модель дискретно-подійного процесу в рамках якого функціонує дана

інформаційна система, побудувати мережу Петрі на основі виявленого процесу та проаналізувати виявлений процес із встановленням таких характеристик процесу:

- частота виконання кожної діяльності;
- частота переходу між кожними виявленими діяльностями;
- тривалість кожної діяльності;
- тривалість затримок (часу між завершенням попередньої діяльності за початком наступної) для кожної діяльності.

3.2 Математична постановка задачі

Сформулюємо математичну модель даної задачі.

Нехай E – набір подій, E* - набір всіх послідовностей, які складаються з нуля або більше подій з E. Тоді $\sigma \in E^*$ це довільна послідовність подій.

$W \subseteq E^*$ – це журнал подій, тобто мультисет послідовностей подій.

Примітка: оскільки W – це мультисет, кожна послідовність подій може з'являтися неодноразово в журналі подій.

Щоб знайти модель процесу на основі журналу подій, журнал необхідно проаналізувати на причинно-наслідкові залежності. Для аналізу цих відносин введемо наступні позначення.

Нехай W буде журналом подій над E, тобто $W \subseteq E^*$.

Нехай $a, b \in E$:

1. $a >_W b$ якщо існує послідовність $\sigma = e_1e_2e_3 \ldots e_n$ та $i \in \{1, \ldots, n-1\}$ такий, що $\sigma \in W$ та $e_i = a$ та $e_i+1 = b$,

2. $a \rightarrow_W b$ якщо $a >_W b$ та не $b >_W a$,

3. $a \#_W b$ якщо не $a >_W$ b та не $b >_W a$,

4. $a ||_W b$ якщо $a >_W b$ та $b >_W a$,

5. $a >>_W b$ якщо існує послідовність $\sigma = e_1e_2e_3 \ldots e_n$ та $i \in \{1, \ldots, n-2\}$ такий, що $\sigma \in W$ та $e_i = a$ та e_i+1 = b та $e_i + 2 = a$,

6. $a >>>_W b$, якщо існує послідовність $\sigma = e_1 e_2 e_3 \ldots e_n$ такий, що $\sigma \in W$ та $e_i = a$ та $e_j = b$.

Нехай $a \Rightarrow_W b$ – показник, заснований на частоті подій, що використовується для позначення того, наскільки ми певні, що існує дійсно залежність між двома подіями A та B. Розраховані $\Rightarrow_W$ значення між подіями журналу подій використовуються в евристичному пошуку правильних відносин залежності [11].

$$a \Rightarrow_w b = \frac{|a>_w b| - |b>_w a|}{|a>_w b| + |b>_w a| + 1} \quad , \tag{3.1}$$

причому $a \Rightarrow_W b \in [-1;1]$.

У випадку, коли $a = b$ маємо:

$$a \Rightarrow_w a = \frac{|a>_w a|}{|a>_w a| + 1} \quad . \tag{3.2}$$

Тепер введемо означення мережі Петрі.

Мережа Петрі є триплетом $N = (P, T, F)$, де P – скінченна множина *позицій*, T - скінчена множина *переходів*, таких, що $P \cap T = \emptyset$, і $F \subseteq (P \times T) \cup (T \times P)$ - сукупність спрямованих дуг [10, 12].

Задача полягає в знаходженні такої мережі Петрі N = (P, T, F), для якої виконується:

$$E_n \subseteq T, \tag{3.3}$$

де $E_n = \{a, b \in E \mid (a >_W b) \geq s \wedge (a \Rightarrow_W b) \geq d\}$, s – порогове значення прямих слідувань, d – порогове значення міри зв'язності.

3.3 Обґрунтування методу розв'язання

Існує багато різних підходів до процесу виявлення дискретно-подійного процесу.

Локальні методи розглядають локальні відносини між подіями в журналах (альфа та евристичний алгоритм), тоді як глобальні підходи будують і уточнюють модель, засновану на цілому журналі (генетичний та нечіткий алгоритм). Евристичний алгоритм використовує частоти та параметризацію

для обробки шуму, в той час як генетичний процес розробки може обробляти складні та шумні журнали, але є дуже ресурсномісткими. Останні підходи зосереджені на управлінні складними реальними моделями або шумними журналами, що використовують кластеризацію та абстракцію, наприклад, на рівні трасування або діяльності.

Алгоритм Альфа (англ. Alpha Miner) – один з перших алгоритмів процесної аналітики, що виявляє мережу робочих процесів (у вигляді мереж Петрі) з журналів подій.

Цей алгоритм застосовує дослідження причинно-наслідкових зв'язків, що спостерігаються між завданнями. Наприклад, одна конкретна подія завжди може передувати іншій конкретній події в кожній з послідовностей виконання, що є корисною і достатньою для даного алгоритму інформацією для виявлення моделі процесу.

Серед обмежень α-алгоритму є його нездатність захоплювати короткі цикли, так звані невидимі події, дублюючі події, конструкції без вільного вибору і шум. Короткі цикли плутаються з паралельними конструкціями при виведенні відносин впорядкування. Конструкції без вільного вибору також не враховуються відносинами впорядкування журналу. Було неодноразово вдосконалено α-алгоритм, але ці вдосконалення все ще мають проблеми при обробці невидимих і дублюючих подій. Крім того, даний алгоритм все ще залишається дуже чутливим до шуму [12].

Евристичний алгоритм (англ. Heuristics Miner) може працювати з шумом і може використовуватися для вираження основної поведінки (тобто не всіх деталей і виключень), зареєстрованих у журналі подій. Він підтримує виявлення всіх поширених конструкцій у моделях процесу (тобто послідовності, вибору, паралелізму, циклів, невидимих завдань і деяких видів невикористання), за винятком дублюючих завдань. Алгоритм має два основних кроки. На першому етапі будується граф залежності. Цей граф залежності містить причинно-наслідкові залежності, які будуть зберігатися

при побудові моделі мережі Петрі. На відміну від алгоритмів, заснованих на абстракції, евристичний алгоритм враховує частоти основних відносин впорядкування під час обчислення сили причинно-наслідкових зв'язків [11, 13].

На другому етапі встановлюється семантика точок розбиття та об'єднання у графі залежності. Використовуючи частоту залежності, вирішується тип розбиття чи об'єднання. Використовуючи порогові значення, приймається рішення, чи має бути вибір або паралельна конструкція в мережі Петрі.

Генетичні алгоритми (англ. Genetic Miner) є адаптивними методами пошуку, які намагаються імітувати процес еволюції. Ці алгоритми починаються з початкової популяції індивідів. Кожному індивіду призначається фітнес-функція для його оцінки. У випадку з виявленням дискретно-подійного процесу, індивідуум є можливим процесом, а придатність - це функція, яка оцінює, наскільки добре він здатен відтворювати поведінку в журналі. Популяції розвиваються шляхом відбору найбільш пристосованих індивідуумів і створення нових за допомогою генетичних операторів, таких як кросовера (об'єднання частин двох або більше осіб) і мутації (випадкова модифікація особи). Крім того, загальноприйнятою практикою є безпосереднє копіювання кількох кращих осіб у поточній популяції (еліти) до наступної популяції. Таким чином, гарантується, що найкращі знайдені особи зберігаються у майбутніх популяціях.

Кожна особина в популяції має внутрішнє уявлення, яке визначає простір пошуку генетичного алгоритму. Разом, внутрішнє уявлення, міра придатності та генетичні оператори складають три основні питання, які необхідно вирішити при розробці генетичних алгоритмів.

Моделі процесу, отримані цим алгоритмом, будуть містити не більше одного завдання для кожної унікальної мітки завдання в журналі. Крім того, заходи фітнесу, що використовуються генетичним алгоритмом, гарантують,

що особи з максимальною придатністю можуть правильно проаналізувати більшість випадків процесу (послідовностей подій) в журналі. Причина цього полягає в тому, що ці алгоритми спрямовані на виявлення моделі, яка відображає якомога ближче поведінку, виражену в журналі подій. Якщо виявлена модель дозволяє створювати багато додаткових послідовностей, які не можуть бути отримані з журналу, вона не дає точного опису того, що насправді відбувається. Іншими словами, ця модель не підходить для журналу. Якщо модель реалізує усі можливі послідовності, вона може переповнювати журнал [14, 15].

Метою використання генетичних алгоритмів є вирішення таких проблем, як дублювання діяльності, приховані діяльності, конструкції невільного вибору, шум і незавершеність, тобто подолання проблем деяких традиційних підходів [16].

Процеси реального життя виявляються менш структурованими, ніж люди, як правило, їх собі уявляють. На жаль, традиційні підходи до видобування технологій мають проблеми, пов'язані з неструктурованими процесами. Знайдені моделі часто є «спагеті-подібними», показуючи всі деталі, не розрізняючи що є важливим, а що ні. **Алгоритм нечіткого виявлення** (англ. Fuzzy Miner) підлаштовується і дозволяє виявляти різні вірно спрощені уявлення конкретного процесу. Такий алгоритм підходить для виявлення менш структурованих процесів, які демонструють велику кількість неструктурованих і конфліктуючих поведінок, тобто для моделей, подібних до так званих спагетті-моделей. Алгоритм нечіткого виявлення використовує різні методи, такі як видалення незначних або нечасто вживаних вузлів, кластеризація високо корельованих вузлів в один вузол і видалення ізольованих вузлів кластерів [3, 17].

Отже, евристичний алгоритм може бути використаний, коли слід працювати з реальними даними з відносно невеликою кількістю різних подій або існує потреба в моделі мережі Петрі для подальшого аналізу

Генетичний алгоритм може бути використаний, мова йдеться про роботу з великою кількістю шуму, обробку дубльованих імен подій, локальними і нелокальними невільними виборами і невидимим подіями.

Алгоритм нечіткого виявлення може бути використаний в переважно для зображення бажаних рис моделі на потрібному рівні абстракції, усунення нерелевантних деталей, зменшення складності та покращення зрозумілості моделі [18].

Беручи до уваги цілі та задачі даного проекту, можна зробити висновок про доцільність застосування саме евристичного алгоритму через необхідність використання виявленої моделі у вигляді мережі Петрі для подальшого її аналізу та можливості використання для імітаційного аналізу в майбутньому.

3.4 Опис методів розв'язання

Евристичний алгоритм (англ. Heuristics Miner) - це алгоритм, що використовується в процесі виявлення моделі. Цей алгоритм фокусується на обчисленні частоти, залежності і послідовності подій при побудові моделі процесу. Для побудови моделі процесу необхідно аналізувати журнали подій на основі значень залежності кожної діяльності [13].

Відправною точкою евристичного алгоритму є побудова так званого графа залежності. Метрика на основі частоти використовується для вказівки того, наскільки ми впевнені, що дійсно існує залежність між двома подіями a і b (позначення $a \Rightarrow_W b$). Розраховані значення $\Rightarrow_W$ між подіями журналу подій використовуються в евристичному пошуку правильних відносин залежності.

Нехай W буде журналом подій над E, тобто $W \subseteq E^*$.

Нехай $a, b \in E$:

1. $a >_W b$ якщо існує послідовність $\sigma = e_1 e_2 e_3 \ldots e_n$ та $i \in \{1, \ldots, n-1\}$ такий, що $\sigma \in W$ та $e_i = a$ та $e_i{+}1 = b$,

2. $a \rightarrow_W b$ якщо $a >_W b$ та не $b >_W a$,

3. $a \#_W b$ якщо не $a >_W$ b та не $b >_W a$,

4. $a \,||_W\, b$ якщо $a >_W b$ та $b >_W a$,

5. $a >>_W b$ якщо існує послідовність $\sigma = e_1e_2e_3 \ldots e_n$ та $i \in \{1, \ldots, n-2\}$ такий, що $\sigma \in W$ та $e_i = a$ та e_i+1 = b та $e_i + 2 = a$,

6. $a>>>_W b$ якщо існує послідовність $\sigma = e_1e_2e_3 \ldots e_n$ такий, що $\sigma \in W$ та $e_i = a$ та $e_j = b$.

Нехай $a \Rightarrow_W b$ – показник, заснований на частоті подій, що використовується для позначення того, наскільки ми певні, що існує дійсно залежність між двома подіями A та B. Розраховані $\Rightarrow_W$ значення між подіями журналу подій використовуються в евристичному пошуку правильних відносин залежності [11].

$$a \Rightarrow_w b = \frac{|a>_w b| - |b>_w a|}{|a>_w b| + |b>_w a| + 1} \quad , \tag{3.4}$$

причому $a \Rightarrow_W b \in [-1;1]$.

У випадку, коли $a = b$ маємо:

$$a \Rightarrow_w a = \frac{|a>_w a|}{|a>_w a| + 1} \ . \tag{3.5}$$

Сформулюємо кроки евристичного алгоритму виявлення моделі процесу.

Крок 1. Розрахунок матриці прямого наслідування.

Формуємо матрицю прямого наслідування на основі кількості зв'язків типу $a >_W b$ для кожних попарно взятих подій $a, b \in E$ у межах кожної окремо взятої послідовності подій σ ∈ E* з журналу подій.

Крок 2. Розрахунок матриці залежності

Формуємо матрицю залежності на основі кількості зв'язків типу $a \Rightarrow_W b$ для кожних попарно взятих подій $a, b \in E$ у межах кожної окремо взятої послідовності подій σ ∈ E* з журналу подій.

Крок 3. Виявлення прийнятних дуг графу залежності

Для кожних попарно взятих подій $a, b \in E$ перевіряємо, чи:

$$|a >_W b| \geq \text{s}, \tag{3.6}$$

$$|a \Rightarrow_W b| \geq d, \tag{3.7}$$

де s – порогове значення прямих слідувань, d – порогове значення міри зв'язності.

Якщо вирази (3.6) та (3.7) виконуються одночасно для двох окремо взятих подій, то це значить, що даний перехід з a до b буде зображений дугою на графі залежності.

Слід перевірити всі значення матриць залежності та прямого слідування для встановлення усіх прийнятних дуг графу залежності. Тобто дуг, що задовольняють встановленим параметрами пороговими значенням.

Набір прийнятних дуг разом з власне подіями, котрі з'єднуються цими дугами, формують граф залежності.

Крок 4. Формування розширеної матриці наслідування

Тепер для встановлення зв'язків розгалуження та сполучення, а отже визначення, чи відбуваються певні подій одночасно або опціонально, слід для кожної окремо взятої послідовності подій $\sigma \in E^*$ з журналу подій та для кожної з прийнятих дуг графу залежності перевірити котрі сусідні події відбулися до аналізованої події, а котрі після. На основі цих даних формується розширена матриця наслідування, що фактично, являє собою табличне зображення графу залежності, доповненого розгалуженнями та з'єднаннями дуг.

Крок 5. Формування мережі Петрі

Маючи граф залежності з розгалуженнями та з'єднаннями, задача побудови мережі Петрі полягає у приведенні утвореного графу до власне мережі Петрі. Сформульована задача вирішується за допомогою власноруч розробленого алгоритму. Таким чином, вершини графу залежності стають переходами мережі Петрі. Позиції та їх зв'язки з переходами мережі Петрі формуються на основі даних про розгалуження та з'єднання розширеної матриці наслідування.

Отримані мережі Петрі точно відображають вхідні дані, на основі яких вони були побудовані – розширену матрицю наслідування. Однак з поставлених задач дипломної роботи відомо, що отримана мережа Петрі має

не тільки функціонально відображати виявлений процес, а й бути інтуїтивно зрозумілою для візуального сприйняття. Тож розроблений алгоритм було модифіковано для можливості отримання якомога простіших зображень мереж Петрі. Це означає, що під час побудови мережі Петрі алгоритм перевіряє, чи не відбувається побудова конструкцій, яка може бути спрощена чи навіть виключена з мережі у конкретному випадку без впливу на функціональні можливості та логіку виявленого процесу. Типовим представником таких конструкцій є конструкція типу «позиція – перехід – позиція – перехід – позиція», що досить очевидно може бути спрощена до «позиція – перехід – позиція». Таким чином, дана модифікація зменшує кількість так званих невидимих (неявних) переходів, зменшує кількість та спрощує структуру складних конструкцій та, отже, робить результуючу мережу Петрі більш простою для візуального сприйняття.

На цьому процес формування мережі Петрі, як, власне, і процес виконання евристичного алгоритму виявлення моделі дискретно-подійного процесу завершується.

Висновок до розділу

У даному розділі було сформульовано змістовну та математичну постановки задач. Окрім цього, було досліджено існуючі методі розв’язання подібних задач (альфа, евристичний, генетичний, нечіткий алгоритми). У результаті було сформульовано висновки про випадки використання кожного з алгоритмів та, враховуючи цілі та задачі проектної роботи, було прийнято рішення про використання саме евристичного алгоритму.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Програмний продукт створювався в середовищі розробки IntelliJ IDEA [19] мовою Java та за допомогою програмного забезпечення для моделювання дискретно-подійних процесів Discrete Event Simulation System [20].

Однією з основних причин, чому Java настільки популярна, є незалежність платформи, що означає, що програми Java можна запускати на багатьох різних типах комп'ютерів. Програма Java працює на будь-якому комп'ютері з інсталяцією Java Runtime Environment, також відомої як JRE. JRE доступний практично для всіх типів комп'ютерів - комп'ютерів під керуванням Windows, комп'ютерів Macintosh, комп'ютерів Unix або Linux, великих комп'ютерів мейнфреймів і навіть мобільних телефонів.

Компілятор Java генерує байт-коди, які не мають нічого спільного з певною комп'ютерною архітектурою, отже, програму Java легко інтерпретувати на будь-якій машині.

Java за своєю природою є об'єктно-орієнтованою, тобто програми Java складаються з елементів програмування, які називаються об'єктами. У Java все являє собою об'єкт, який має деякі дані і поведінку. Java може бути легко розширена, оскільки заснована на об'єктній моделі.

Java намагається усунути схильні до помилок ситуації, наголошуючи головним чином на перевірці помилок під час компіляції та перевірці під час виконання.

З багатопоточністю Java можна писати програми, які можуть виконувати багато завдань одночасно. Ця функція дозволяє розробникам створювати інтерактивні програми, які можуть працювати безперебійно. Головною перевагою багатопоточності є те, що вона не займає пам'ять для кожного потоку. Вона розподіляє спільну область пам'яті. Потоки дуже корисні та

важливі при розробці продуктів, пов’язаних з мультимедіа, веб-додатками тощо [21].

IntelliJ IDEA є спеціальним середовищем програмування або інтегрованим середовищем розробки (IDE), багато в чому призначеним для Java. Це середовище використовується для розробки програм. Розроблене компанією JetBrains. IntelliJ IDEA представлена у двох виданнях: Community Edition, який ліцензується Apache 2.0, і комерційне видання, відоме як Ultimate Edition. Що відрізняє IntelliJ IDEA від своїх аналогів – це простота використання, гнучкість і міцний дизайн.

IntelliJ IDEA являє собою високотехнологічний комплекс тісно інтегрованих інструментів програмування, що включає інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації, потужні інструменти рефакторинга коду, вбудовану підтримку технологій J2EE, механізми інтеграції з середовищем тестування Ant / JUnit і системами управління версіями, унікальний інструмент оптимізації та перевірки коду Code Inspection, а також інноваційний візуальний конструктор графічних інтерфейсів [19].

У доповіді Infoworld 2010, IntelliJ отримала найвищу оцінку тестового центру серед чотирьох провідних інструментів програмування Java: Eclipse, IntelliJ IDEA, NetBeans та JDeveloper.

Discrete Event Simulation System – зручний у користуванні компонент візуального програмування стохастичних мереж Петрі, що дозволяє будувати стохастичні мережі Петрі з багатоканальними переходами та інформаційними зв’язками. Це вдається здійснювати шляхом маніпуляцій з графічними об’єктами (позиціями, переходами, вхідними та вихідними дугами). Окрім цього даний компонент візуального програмування надає можливість зберігати, відкривати, модифікувати збережені моделі, моделювати поведінку відповідних систем у часі та збирати статистичні дані щодо їхнього функціонування [22].

4.2 Вимоги до технічного забезпечення

Для коректної роботи даного програмного продукту до технічного забезпечення мають входити:

1. комп’ютер:

1.1. з процесором з тактовою частотою не нижче 1 ГГц;

1.2. з об’ємом оперативної пам’яті не менше 1 ГБ;

2. програмне забезпечення:

2.1. операційна система Windows 7 та вище;

2.2. Java SE Runtime Environment 8 та вище;

3. комп’ютерна периферія:

3.1. монітор;

3.2. мишка.

4.3 Архітектура програмного забезпечення

У ході розробки програмного продукту було побудовано діаграму класів та діаграму послідовності, доцільність яких сформульовано та описано у таблиці 4.1.

Таблиця 4.1 – Доцільність побудови діаграм

Діаграма	Роль
Діаграма класів (Class diagram)	У процесі проектування визначає сукупність класів, їх структуру та взаємозв’язки між ними для реалізації певної функції чи вирішення певної задачі. Надає змогу оцінити та прийняти відповідні рішення щодо структурної організації програмного продукту

Продовження таблиці 4.1

Діаграма послідовності (Sequence diagram)	У процесі проектування, по-перше, визначає те, як класи об’єднуються у компоненти, та взаємозв’язок цих компонент. По-друге, демонструє послідовність викликів кожного з компонентів під час роботи програмного продукту. По-третє, дає чітке та структуроване розуміння процесу виконання деякої частини програмного продукту або, навіть, його у цілому

4.3.1 Діаграма класів

На діаграмі класів розробленого програмного продукту (схему структурну класів наведено в графічних матеріалах) зображено класи з їх структурою та взаємозв’язками, що реалізують такі функції як зчитування вхідного файлу журналу подій у вигляді файлу з розширенням .csv, обробка вхідних даних, задання параметрів виявлення дискретно-подійного процесу, виявлення дискретно-подійного процесу за допомогою евристичного алгоритму процесної аналітики на основі журналу подій та параметрів виявлення, побудова мережі Петрі на основі виявленого дискретно-подійного процесу, графічна візуалізація виявленого дискретно-подійного процесу, аналіз дискретно-подійного процесу на основі даних журналу подій та графічна візуалізація результатів аналізу виявленого процесу у вигляді інтерактивних таблиць.

Логіку програмного продукту було реалізовано з використанням 9 класів: Log, LogItem, Node, Arc, InputData, IntPair, HeuristicMiner, ReportFrame та ProcessMiningFrame.

ProcessMiningFrame – головний клас програмного продукту, що реалізує графічний інтерфейс введення вхідних даних, графічну візуалізацію введених вхідних даних, введення параметрів виявлення дискретно-подійного

процесу, перевірку коректності введених даних і параметрів та містить функції, що ініціюють власне виявлення дискретно-подійного процесу методами процесної аналітики (Process Mining) та графічну візуалізацію як мережі Петрі, побудованої на основі виявленого процесу, так і результатів аналізу даного процесу.

ReportFrame – клас програмного продукту, що реалізує графічну візуалізацію результатів аналізу виявленого дискретно-подійного процесу у вигляді інтерактивних таблиць з даними з можливістю довільного впорядкування та сортування вмісту.

HeuristicMiner – клас, що реалізує у собі математичну модель програмного продукту, що включає в себе обробку вхідних даних, виявлення дискретно-подійного процесу за допомогою евристичного алгоритму, аналіз виявленого процесу та формування вихідних даних.

InputData – клас, необхідний для збереження та роботи з вхідними даними та заданими параметрами у потрібному форматі.

Log – клас, що зберігає у собі зчитані та оброблені вхідні дані – журнал подій.

LogItem – клас, що використовується як структурний елемент класу Log, та зберігає зчитаний та оброблений рядок журналу подій.

Node – клас, що задає структуру кожного з елементів-вершин графу залежності, необхідного для виявлення дискретно-подійного процесу.

Arc – клас, що задає структуру кожного з елементів-дуг графу залежності, необхідного для виявлення дискретно-подійного процесу.

IntPair – допоміжний клас, що слугує для збереження та роботи з парами цілих чисельних величин типу Integer.

4.3.2 Діаграма послідовності

Діаграма послідовності, наведена в графічних матеріалах, демонструє приклад типового використання програмного продукту користувачем:

починаючи з введення вхідних даних та завершуючи переглядом графічної візуалізації виявленого дискретно-подійного процесу у вигляді мережі Петрі та результатів аналізу даного процесу у вигляді інтерактивного звіту у табличному поданні.

Опис задіяних на діаграмі послідовності компонентів та їх відповідальності наведено в таблиці 4.2.

Таблиця 4.2 – Опис компонентів діаграми послідовності

Компонент	Відповідальність
Користувач	Введення вхідних даних, введення параметрів виявлення процесу, перегляд візуалізованої мережі Петрі, перегляд візуалізованого звіту з аналізу виявленого процесу
Інтерфейс	Відображення форми вибору файлу, відображення форми введення вхідних даних, відображення форми введення параметрів виявлення процесу, відображення графічної візуалізації мережі Петрі, відображення графічної візуалізації звіту з аналізу виявленого процесу
Математична модель	Обробка вхідних даних, виявлення дискретно-подійного процесу за допомогою евристичного алгоритму, аналіз дискретно-подійного процесу за даними журналу подій, формування вихідних даних

4.3.3 Специфікація функцій

Функції класів розробленого програмного продукту наведено та описано в таблиці 4.3.

Таблиця 4.3 – Специфікація функцій

Назва	Примітка
Клас **ProcessMiningFrame** – головний клас програмного продукту, що реалізує графічний інтерфейс введення та виведення інформації	
private void init(List<List<String>> list)	Ініціалізує форму введення вхідних даних та візуалізацію введених даних у табличному поданні
private JPanel getNextParametersComponents()	Ініціалізує форму введення додаткових параметрів виявлення процесу
private JPanel getTracesInfoComponents(InputData inputData)	Зображує вхідні дані у вигляді таблиці, що містить назви, послідовності подій та їх частоту
Клас **ReportFrame** – клас програмного продукту, що реалізує графічну візуалізацію результатів аналізу виявленого дискретно-подійного процесу	
private void init(HeuristicMiner miner)	Ініціалізує форму візуалізації отриманих результатів аналізу виявленого процесу
public double round(double value, int places)	Округлює числове значення типу double до певної кількості знаків після крапки
Клас **HeuristicMiner** – клас, що реалізує у собі математичну модель програмного продукту	
public PetriNet runConsecutively(InputData inputData)	Виявлення дискретно-подійного процесу на основі журналу подій та параметрів виявлення за допомогою евристичного алгоритму процесної аналітики.

Продовження таблиці 4.3

public void analyse(InputData inputData, Log log, int[][] directSuccessionMatrix, double[][] dependencyMeasuresMatrix, ArrayList<Arc> approvedArcs, ArrayList<Node> nodes)	Аналіз виявленого дискретно-подійного процесу, що включає виявлення частоти виявлених подій, їх тривалості та затримок між ними
public int[][] calculateDirectSuccessionMatrix(Log log)	Розраховує матрицю прямих наслідувань, необхідну для виявлення моделі процесу
private double[][] calculateDependencyMeasuresMatrix(int[][] successionMatrix)	Розраховує матрицю міри зв’язності, необхідну для виявлення моделі процесу
private ArrayList<Arc> getApprovedArcs(Log log, int[][] directSuccessionMatrix, int successionsThreshold, double[][] dependencyMeasuresMatrix, double dependencyThreshold)	Формує список дуг графу залежності, що задовольняють вимогам параметрів виявлення процесу
private ArrayList<Node> getNodes(Log log, ArrayList<Arc> approvedArcs, int windowSize)	Формує список вершин графу залежності, що задовольняють вимогам параметрів виявлення процесу

Продовження таблиці 4.3

private PetriNet generatePetriNet_simplified(ArrayList<Arc> arcs, ArrayList<Node> nodes)	Створює необхідні дані, для ініціалізації та візуалізації мережі Петрі, побудованої на основі виявленого процесу
private void newPoint(String title, int tokens)	Додає нову позицію мережі Петрі
private void newTransition(String title, double time)	Додає новий перехід мережі Петрі
private void newInArc(int n, int m)	Додає новий зв’язок між позицією та переходом мережі Петрі
private void newOutArc(int n, int m)	Додає новий зв’язок між переходом та позицією мережі Петрі
Клас **Log** – клас, що зберігає у собі зчитані та оброблені вхідні дані – журнал подій	
public void add(LogItem logItem)	Додає новий запис до журналу подій
Клас **LogItem** – клас, що використовується як структурний елемент класу Log, та зберігає зчитаний та оброблений рядок журналу подій	
public ArrayList<String> getTrace()	Повертає список послідовностей подій журналу подій
public String toString()	Повертає рядкове подання даного об’єкту
Клас **InputData** – клас, необхідний для збереження та роботи з вхідними даними та заданими параметрами у потрібному форматі	
public String initLog()	Ініціалізує журнал подій за вхідними даними у табличному поданні
public void initActivityFullToShortTitle()	Ініціалізує словник, що встановлює відповідність між повною та скороченою назвою події

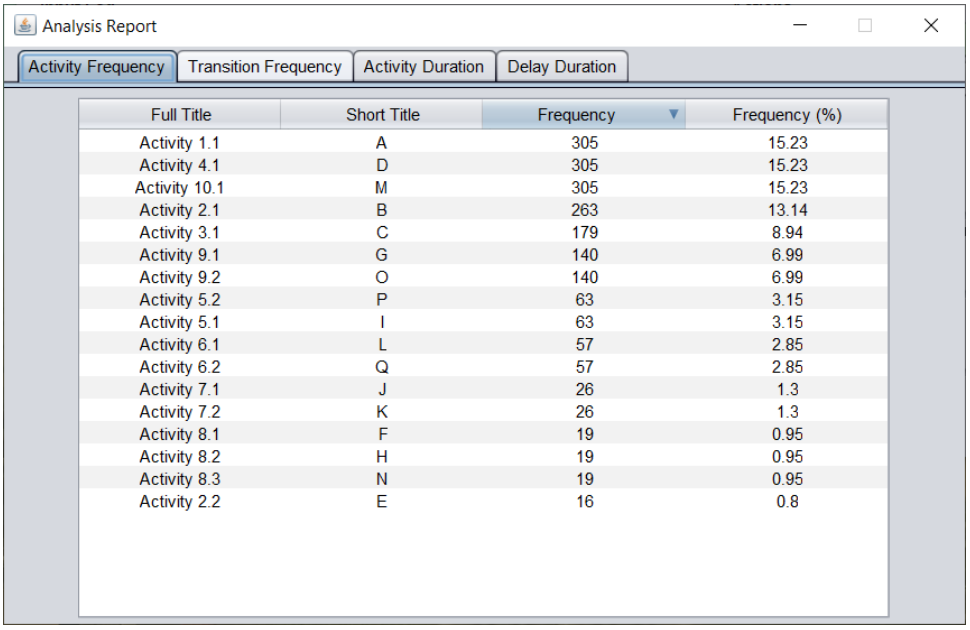
Продовження таблиці 4.3

Клас **Node** – клас, що задає структуру кожного з елементів-вершин графу залежності, необхідного для виявлення дискретно-подійного процесу	
public String toString()	Повертає рядкове подання даного об’єкту
public void incrementFrequency(int n)	Збільшує частоту виявлення даної події на значення параметра n
public void addInputBindings(HashMap<HashSet<Integer>,Integer> add)	Додає вхідні переходи до даної вершини графу залежності
public void addOutputBindings(HashMap<HashSet<Integer>,Integer> add)	Додає вихідні переходи до даної вершини графу залежності
Клас **Arc** – клас, що задає структуру кожного з елементів-дуг графу залежності, необхідного для виявлення дискретно-подійного процесу	
public String toString()	Повертає рядкове подання даного об’єкту
Клас **IntPair** – допоміжний клас, що слугує для збереження та роботи з парами цілих чисельних величин типу Integer	
public boolean equals(Object v)	Використовується для перевірки на ідентичність двох об’єктів даного класу
public int hashCode()	Використовується для формування унікального ідентифікатору для кожного об’єкта даного класу

4.4 Опис звітів

Запропонований програмний продукт надає можливість збору статистичної інформації про виявлений процес та формування на основі цих даних інтерактивного звіту. Інтерактивний звіт складається з чотирьох таблиць, стовпці та рядки яких вільно сортуються та впорядковуються з розсуду користувача для зручності їх перегляду та сприйняття.

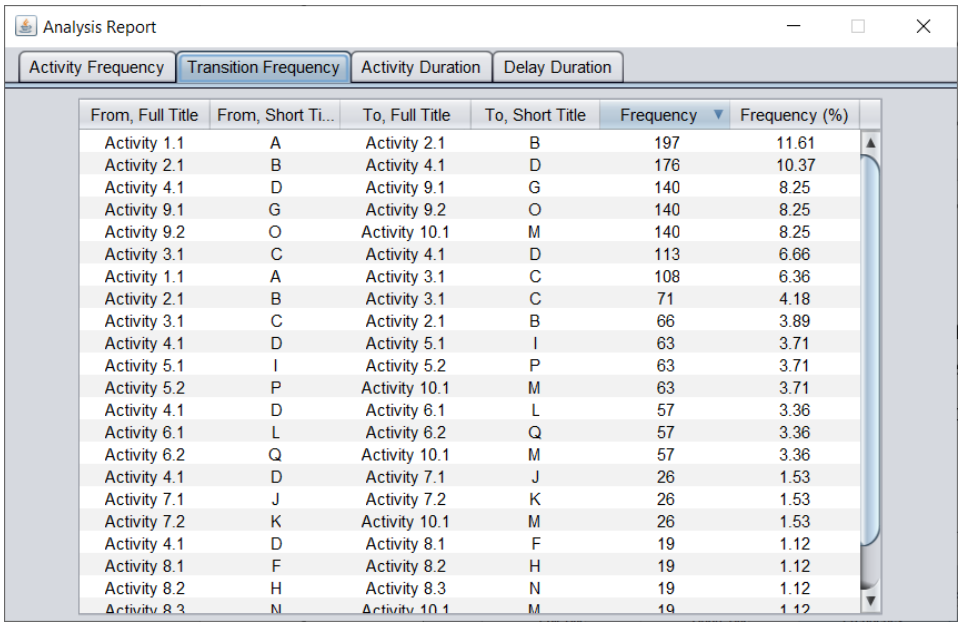
Приклад сформованого звіту та описи побудованих у ньому таблиць зображено на рисунках 4.1-4.4.



Full Title	Short Title	Frequency	Frequency (%)
Activity 1.1	A	305	15.23
Activity 4.1	D	305	15.23
Activity 10.1	M	305	15.23
Activity 2.1	B	263	13.14
Activity 3.1	C	179	8.94
Activity 9.1	G	140	6.99
Activity 9.2	O	140	6.99
Activity 5.2	P	63	3.15
Activity 5.1	I	63	3.15
Activity 6.1	L	57	2.85
Activity 6.2	Q	57	2.85
Activity 7.1	J	26	1.3
Activity 7.2	K	26	1.3
Activity 8.1	F	19	0.95
Activity 8.2	H	19	0.95
Activity 8.3	N	19	0.95
Activity 2.2	E	16	0.8

Рисунок 4.1 – Таблиця «Частота подій» сформованого звіту

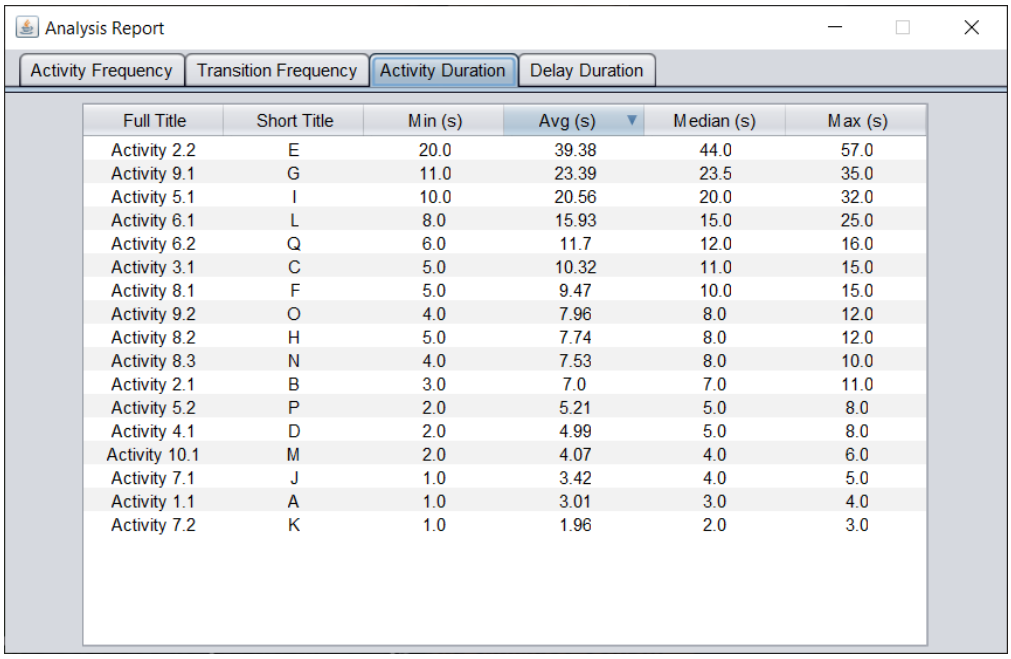
Розділ звіту «Частота подій» (англ. Activity Frequency) (рисунок 4.1) надає інформацію про частоту надання кожної з виявлених подій та містить чотири стовпці: Повна назва події (англ. Full Title), скорочене позначення події (англ. Short Title), частота настання події (англ. Frequency) та частота настання події у відсотках (англ. Frequency (%)).



From, Full Title	From, Short Ti...	To, Full Title	To, Short Title	Frequency	Frequency (%)
Activity 1.1	A	Activity 2.1	B	197	11.61
Activity 2.1	B	Activity 4.1	D	176	10.37
Activity 4.1	D	Activity 9.1	G	140	8.25
Activity 9.1	G	Activity 9.2	O	140	8.25
Activity 9.2	O	Activity 10.1	M	140	8.25
Activity 3.1	C	Activity 4.1	D	113	6.66
Activity 1.1	A	Activity 3.1	C	108	6.36
Activity 2.1	B	Activity 3.1	C	71	4.18
Activity 3.1	C	Activity 2.1	B	66	3.89
Activity 4.1	D	Activity 5.1	I	63	3.71
Activity 5.1	I	Activity 5.2	P	63	3.71
Activity 5.2	P	Activity 10.1	M	63	3.71
Activity 4.1	D	Activity 6.1	L	57	3.36
Activity 6.1	L	Activity 6.2	Q	57	3.36
Activity 6.2	Q	Activity 10.1	M	57	3.36
Activity 4.1	D	Activity 7.1	J	26	1.53
Activity 7.1	J	Activity 7.2	K	26	1.53
Activity 7.2	K	Activity 10.1	M	26	1.53
Activity 4.1	D	Activity 8.1	F	19	1.12
Activity 8.1	F	Activity 8.2	H	19	1.12
Activity 8.2	H	Activity 8.3	N	19	1.12
Activity 8.3	N	Activity 10.1	M	19	1.12

Рисунок 4.2 – Таблиця «Частота переходів» сформованого звіту

Розділ звіту «Частота переходів» (англ. Transition Frequency) (рисунок 4.2) надає інформацію про частоту переходів між двома будь-якими подіями виявленого процесу та містить шість стовпців: Звідки, повна назва події (англ. From, Full Title); звідки, скорочене позначення події (англ. From, Short Title); куди, повна назва події (англ. To, Full Title); куди, скорочене позначення події (англ. To, Short Title); частота переходу між подіями (англ. Frequency) та частота переходу між подіями у відсотках (англ. Frequency (%)).



Full Title	Short Title	Min (s)	Avg (s)	Median (s)	Max (s)
Activity 2.2	E	20.0	39.38	44.0	57.0
Activity 9.1	G	11.0	23.39	23.5	35.0
Activity 5.1	I	10.0	20.56	20.0	32.0
Activity 6.1	L	8.0	15.93	15.0	25.0
Activity 6.2	Q	6.0	11.7	12.0	16.0
Activity 3.1	C	5.0	10.32	11.0	15.0
Activity 8.1	F	5.0	9.47	10.0	15.0
Activity 9.2	O	4.0	7.96	8.0	12.0
Activity 8.2	H	5.0	7.74	8.0	12.0
Activity 8.3	N	4.0	7.53	8.0	10.0
Activity 2.1	B	3.0	7.0	7.0	11.0
Activity 5.2	P	2.0	5.21	5.0	8.0
Activity 4.1	D	2.0	4.99	5.0	8.0
Activity 10.1	M	2.0	4.07	4.0	6.0
Activity 7.1	J	1.0	3.42	4.0	5.0
Activity 1.1	A	1.0	3.01	3.0	4.0
Activity 7.2	K	1.0	1.96	2.0	3.0

Рисунок 4.3 – Таблиця «Тривалість подій» сформованого звіту

Розділ звіту «Тривалість подій» (англ. Activity Duration) (рисунок 4.3) надає інформацію про статистичні оцінки тривалості кожної з виявлених подій та містить шість стовпців: Повна назва події (англ. Full Title), скорочене позначення події (англ. Short Title), мінімальна тривалість події в секундах (Min (s)), середня арифметична тривалість події в секундах (Avg (s)), медіанна тривалість події в секундах (Median (s)), максимальна тривалість події в секундах (Max (s)).

From, Full...	From, Sh...	To, Full Ti...	To, Short ...	Min (s)	Avg (s)	Median (s)	Max (s)
Activity 9.2	O	Activity 1...	M	4.0	22.75	23.5	42.0
Activity 5.2	P	Activity 1...	M	4.0	20.92	20.0	37.0
Activity 6.2	Q	Activity 1...	M	5.0	17.67	17.0	30.0
Activity 4.1	D	Activity 6.1	L	1.0	12.98	9.0	67.0
Activity 8.2	H	Activity 8.3	N	3.0	11.16	12.0	17.0
Activity 4.1	D	Activity 7.1	J	2.0	9.35	8.5	42.0
Activity 4.1	D	Activity 8.1	F	2.0	9.11	8.0	30.0
Activity 4.1	D	Activity 5.1	I	2.0	9.1	7.0	66.0
Activity 4.1	D	Activity 9.1	G	1.0	8.48	9.0	20.0
Activity 8.3	N	Activity 1...	M	2.0	7.95	8.0	14.0
Activity 2.1	B	Activity 4.1	D	0.0	5.77	4.0	18.0
Activity 3.1	C	Activity 4.1	D	1.0	5.69	5.0	13.0
Activity 2.2	E	Activity 4.1	D	1.0	5.69	7.0	11.0
Activity 6.1	L	Activity 6.2	Q	1.0	5.23	6.0	9.0
Activity 8.1	F	Activity 8.2	H	1.0	4.89	5.0	9.0
Activity 5.1	I	Activity 5.2	P	1.0	4.75	5.0	9.0
Activity 9.1	G	Activity 9.2	O	1.0	4.43	4.0	9.0
Activity 7.1	J	Activity 7.2	K	1.0	4.38	4.0	9.0
Activity 1.1	A	Activity 2.1	B	0.0	4.05	4.0	9.0
Activity 1.1	A	Activity 3.1	C	1.0	3.84	4.0	7.0
Activity 2.1	B	Activity 3.1	C	0.0	3.18	3.0	5.0
Activity 3.1	C	Activity 2.1	B	0.0	2.94	3.0	6.0

Рисунок 4.4 – Таблиця «Тривалість затримок» сформованого звіту

Розділ звіту «Тривалість затримок» (англ. Delay Duration) (рисунок 4.4) надає інформацію про статистичні оцінки тривалості затримок між будь-якими двома подіями виявленого процесу та містить вісім стовпців: Звідки, повна назва події (англ. From, Full Title); звідки, скорочене позначення події (англ. From, Short Title); куди, повна назва події (англ. To, Full Title); куди, скорочене позначення події (англ. To, Short Title); мінімальна тривалість затримки між подіями в секундах (Min (s)), середня арифметична тривалість затримки між подіями в секундах (Avg (s)), медіанна тривалість затримки між подіями в секундах (Median (s)), максимальна тривалість затримки між подіями в секундах (Max (s)).

Висновок до розділу

У даному розділі було сформульовано та аргументовано доцільність побудови діаграм класів та послідовності до розроблюваного програмного продукту. Було побудовано та детально описано дані діаграми, наведено специфікацію функцій у табличному поданні. Окрім цього, було

продемонстровано та описано звіти, створені на основі результатів аналізу виявленого дискретно-подійного процесу розробленим програмним продуктом.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Робота з програмним продуктом починається з його запуском. Для цього можна:

- скористатися графічним інтерфейсом операційної системи та клікнути на файл запуску програмного продукту DiscreteEventSimulationSystem.jar;
- скористатися консольною командою java -jar DiscreteEventSimulationSystem.jar.

Після запуску відкриється головне вікно компоненту візуального програмування стохастичних мереж Петрі (рисунок 5.1).

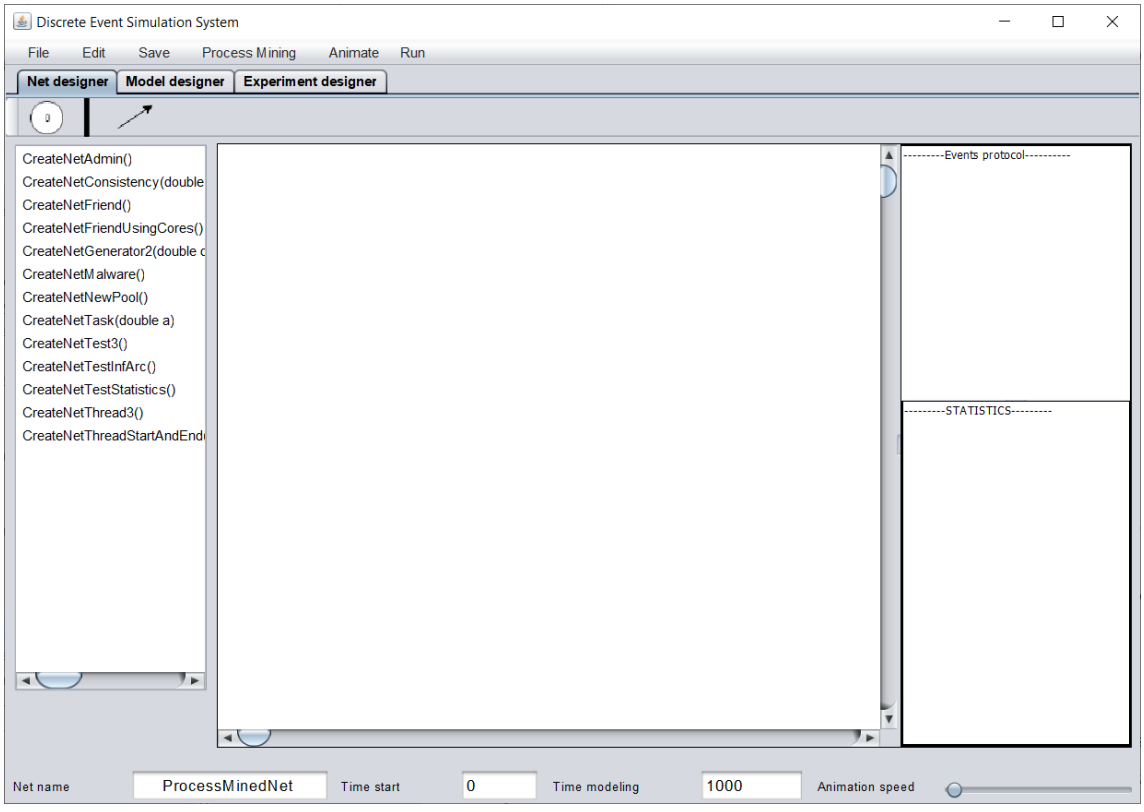


Рисунок 5.1 – Головне вікно Discrete Event Simulation System

Для виявлення дискретно-подійного процесу за допомогою евристичного алгоритму виявлення процесу методами процесної аналітики (англ. Process Mining) слід на головному меню програмного продукту обрати

пункт Process Mining та обрати підпункт завантаження вхідних даних (Open Event Data) (рисунок 5.2).

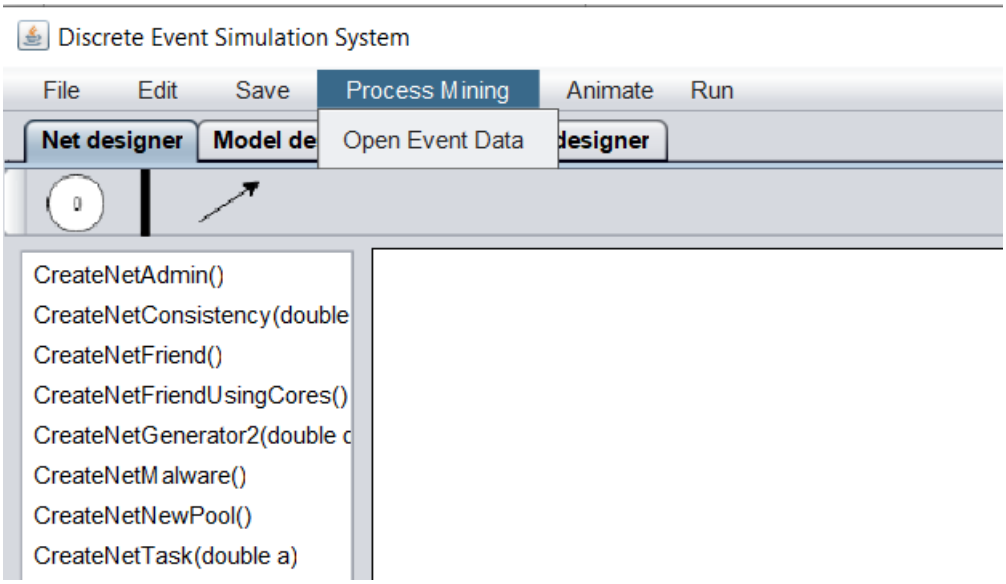


Рисунок 5.2 – Головне меню з пунктом Process Mining та підпунктом Open Event Data

У відкритому діалоговому вікні слід знайти та обрати файл для завантаження. Вхідних файл має містити записи про виконання подій у вигляді власне журналу подій. Вхідний файл має бути розширення *.csv. На рисунку 5.3 зображено завантаження файлу event_log.csv.

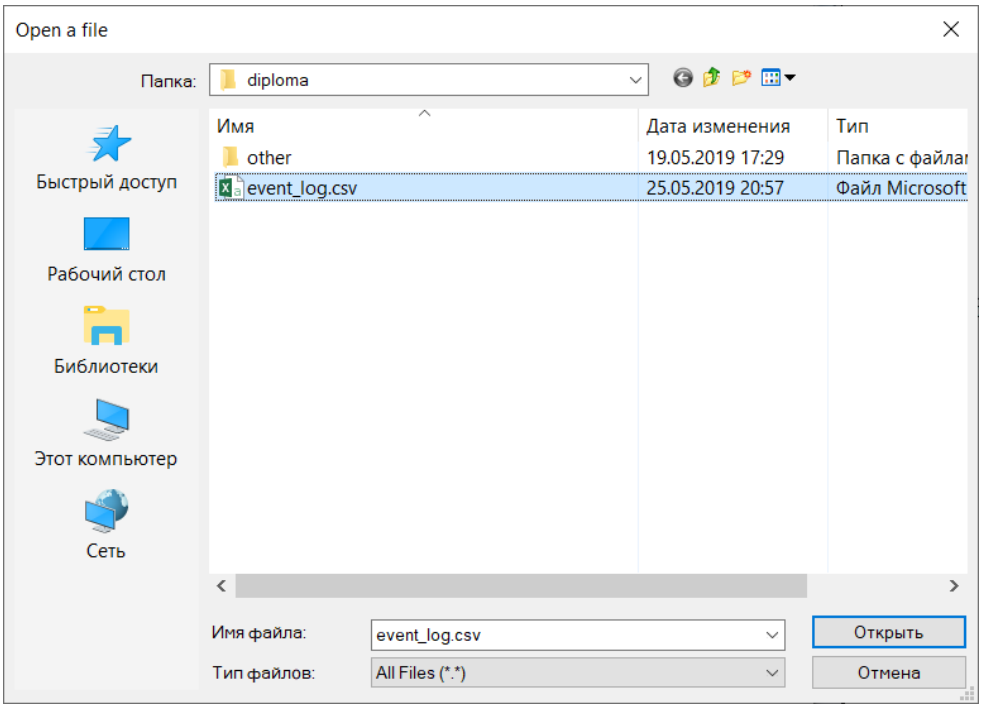


Рисунок 5.3 – Вікно завантаження вхідних даних – журналу подій

Після завантаження вхідних даних з’являється вікно перегляду журналу подій (рисунок 5.4).

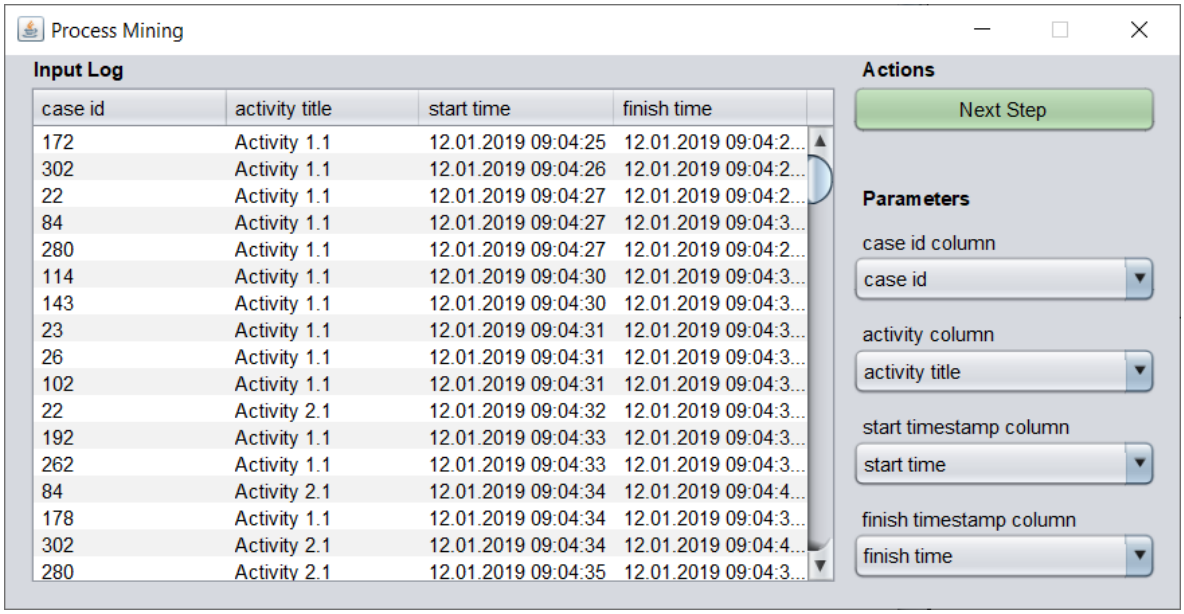


Рисунок 5.4 – Вікно перегляду журналу подій

Тепер слід переглянути завантажену таблицю, знайти праворуч поля введення інформації (параметри у вигляді випадаючих списків), та вказати назви стовпці журналу подій, що відповідають потрібним для аналізу процесу даних: ідентифікатори випадків (case id), назви подій (activity), часові мітки початку події (start timestamp), часові мітки завершення подій (finish timestamp). Приклад коректного заповнення цих даних зображено на рисунку 5.5.

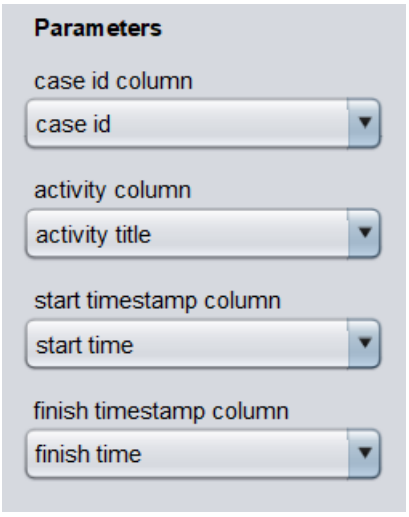


Рисунок 5.5 – Коректне задання параметрів

Після заповнення даних слід натиснути на кнопку Next Step для переходу на наступний крок завантаження вхідних даних.

На наступному кроці (рисунок 5.6) слід задати параметри виявлення процесу: порогове значення кількості прямих наслідувань (succession threshold), порогове значення міри зв'язності (dependency threshold), значення розміру вікна для встановлення розгалужень та з'єднань (window size), та зазначити необхідність підготовки процесу до моделювання (створення додаткових елементів моделі, що створить можливість проведення імітаційного моделювання засобами Discrete Event Simulation System). При відсутності такої необхідності результуюча модель не отримає вище згаданих додаткових елементів та відповідно буде простішою для візуального сприйняття.

Рисунок 5.6 – Вікно Process Mining на другому кроці

Параметри за замовчуванням задано з урахуванням завантаженого журналу подій та належать до діапазону допустимих значень.

Після задання параметрів, слід натиснути на кнопку Generate Petri Net для виявлення дискретно-подійного процесу за вхідними даними та параметрами та його візуалізації у вигляді мережі Петрі. Мережу Петрі можна спостерігати у головному вікні програмного застосунку (рис 5.7).

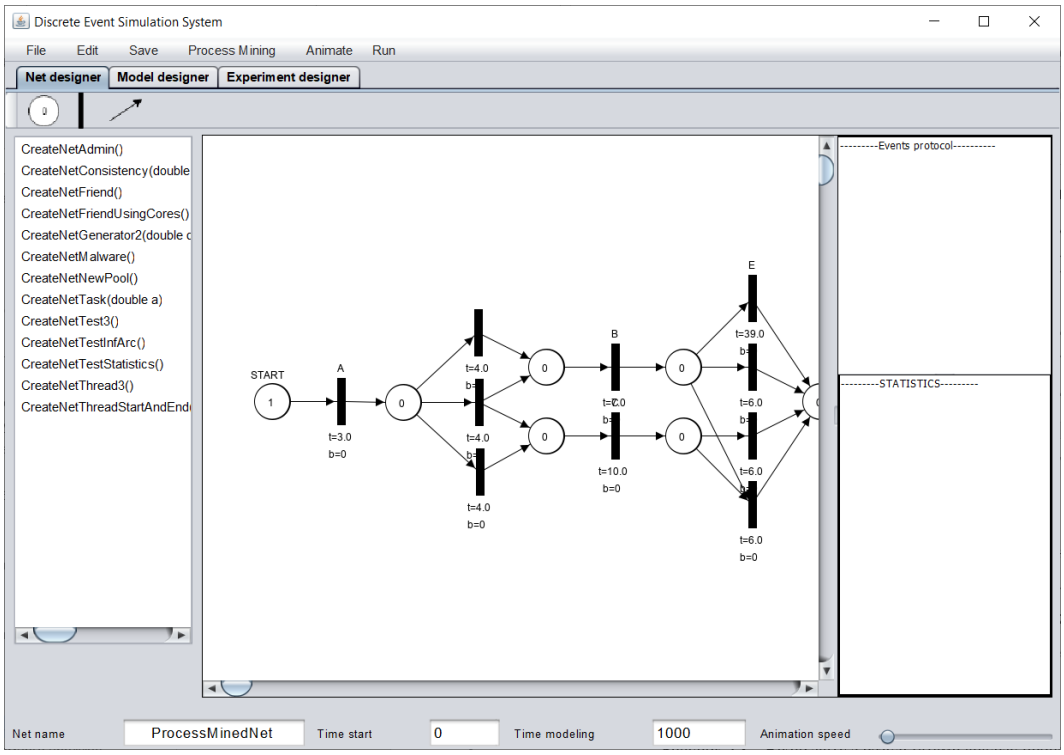
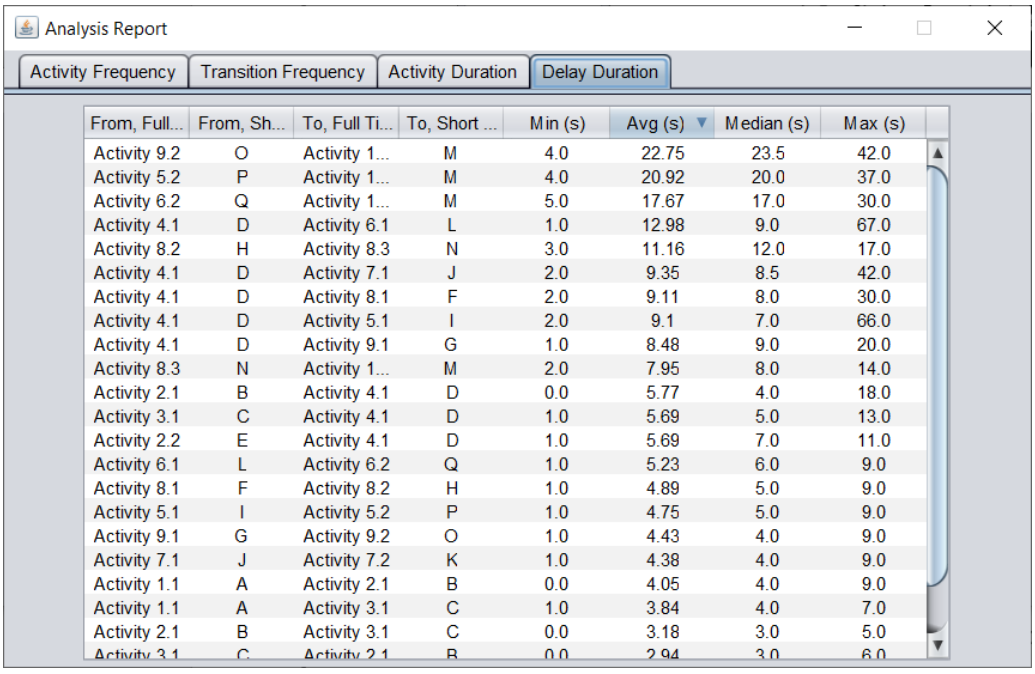


Рисунок 5.7 – Головне вікно програмного продукту з візуалізованим процесом

Для формування та перегляду звіту виявленого процесу слід натиснути на кнопку Generate Report вікна Process Mining. Після чого відкриється вікно звіту (рисунок 5.8), де на кожній з чотирьох вкладок представлені результати аналізу процесу у вигляді таблиць.



From, Full...	From, Sh...	To, Full Ti...	To, Short ...	Min (s)	Avg (s)	Median (s)	Max (s)
Activity 9.2	O	Activity 1...	M	4.0	22.75	23.5	42.0
Activity 5.2	P	Activity 1...	M	4.0	20.92	20.0	37.0
Activity 6.2	Q	Activity 1...	M	5.0	17.67	17.0	30.0
Activity 4.1	D	Activity 6.1	L	1.0	12.98	9.0	67.0
Activity 8.2	H	Activity 8.3	N	3.0	11.16	12.0	17.0
Activity 4.1	D	Activity 7.1	J	2.0	9.35	8.5	42.0
Activity 4.1	D	Activity 8.1	F	2.0	9.11	8.0	30.0
Activity 4.1	D	Activity 5.1	I	2.0	9.1	7.0	66.0
Activity 4.1	D	Activity 9.1	G	1.0	8.48	9.0	20.0
Activity 8.3	N	Activity 1...	M	2.0	7.95	8.0	14.0
Activity 2.1	B	Activity 4.1	D	0.0	5.77	4.0	18.0
Activity 3.1	C	Activity 4.1	D	1.0	5.69	5.0	13.0
Activity 2.2	E	Activity 4.1	D	1.0	5.69	7.0	11.0
Activity 6.1	L	Activity 6.2	Q	1.0	5.23	6.0	9.0
Activity 8.1	F	Activity 8.2	H	1.0	4.89	5.0	9.0
Activity 5.1	I	Activity 5.2	P	1.0	4.75	5.0	9.0
Activity 9.1	G	Activity 9.2	O	1.0	4.43	4.0	9.0
Activity 7.1	J	Activity 7.2	K	1.0	4.38	4.0	9.0
Activity 1.1	A	Activity 2.1	B	0.0	4.05	4.0	9.0
Activity 1.1	A	Activity 3.1	C	1.0	3.84	4.0	7.0
Activity 2.1	B	Activity 3.1	C	0.0	3.18	3.0	5.0
Activity 3.1	C	Activity 2.1	B	0.0	2.94	3.0	6.0

Рисунок 5.8 – Вікно звіту з результатами аналізу процесу

Після перегляду мережі Петрі та результатів аналізу виявленого процесу користування програмним продуктом завершується.

5.2 Випробування програмного продукту

У даному підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності розробленого програмного продукту функціональним вимогам, представленим у технічному завданні на створення програмного продукту для аналізу та моделювання дискретно-подійного процесу на основі журналу подій.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій програмного продукту для аналізу та моделювання дискретно-подійного процесу на основі журналу подій вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

У процесі тестування програмного продукту (ПП) була перевірена уся його функціональність, зазначена в ТЗ.

У таблицях 5.1-5.12 наведено перелік, опис та хід випробувань основних функціональних можливостей розробленого програмного продукту.

Таблиця 5.1 – Відкриття вікна завантаження вхідних даних

Мета тесту:	**Перевірка функції «Завантаження вхідних даних»**
Початковий стан ПП	Відкрите головне вікно ПП
Вхідні дані:	Файл журналу подій розширення *.csv
Схема проведення тесту:	Обрати підпункт Open Event Data пункту Process Mining з головного меню
Очікуваний результат:	Відкрите вікно вибору файлу Open a file
Стан ПП після проведення випробувань:	Відкрите вікно вибору файлу Open a file

Таблиця 5.2 – Завантаження вхідних даних

Мета тесту:	**Перевірка функції «Завантаження вхідних даних»**
Початковий стан ПП	Відкрите вікно вибору файлу Open a file
Вхідні дані:	Файл журналу подій з розширенням *.csv
Схема проведення тесту:	Обрати файл журналу подій з розширенням *.csv для завантаження. Натиснути кнопку «Відкрити»
Очікуваний результат:	Відкрите вікно перегляду журналу подій
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 5.3 – Вибір стовпця ідентифікатора випадків

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою activity title
Схема проведення тесту:	Визначити параметр case id column як activity title. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обраний стовпець ідентифікатора подій
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 5.4 – Вибір стовпця мітки часу початку

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою activity title
Схема проведення тесту:	Визначити параметр start timestamp column як activity title. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обраний стовпець мітки часу початку
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 5.5 – Вибір стовпця мітки часу завершення

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою activity title
Схема проведення тесту:	Визначити параметр finish timestamp column як activity title. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обраний стовпець мітки часу завершення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 5.6 – Вибір стовпця міток часу початку та завершення

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою start time, finish time
Схема проведення тесту:	Визначити параметр start timestamp column як finish time, а параметр finish timestamp column як start time. Натиснути кнопку Next Step

Продовження таблиці 5.6

Очікуваний результат:	Повідомлення про некоректно обрані стопці міток часу початку та завершення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 5.7 – Задання параметрів виявлення процесу

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою case id, activity title, start time, finish time
Схема проведення тесту:	Визначити параметр case id column як case id; параметр activity column як activity title, start timestamp column як start time, а параметр finish timestamp column як finish time. Натиснути кнопку Next Step
Очікуваний результат:	Перехід до наступного кроку виявлення. Зображення таблиць назв діяльності та її скорочення; послідовність подій та їх частота, нові параметри для введення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій на другому кроці

Таблиця 5.8 – Виявлення дискретно подійного-процесу 1

Мета тесту:	**Перевірка функції «Виявлення дискретно подійного-процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій на другому кроці
Вхідні дані:	Значення параметру succession threshold = 197 (максимум повзунку)
Схема проведення тесту:	Визначити параметр succession threshold = 197. Натиснути кнопку Generate Petri net

Продовження таблиці 5.8

Очікуваний результат:	Повідомлення про помилку виявлення процесу, так як виявлена мережа незв'язна. Рекомендація змінити параметри виявлення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій на другому кроці

Таблиця 5.9 – Виявлення дискретно подійного-процесу 2

Мета тесту:	**Перевірка функції «Виявлення дискретно подійного-процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій на другому кроці
Вхідні дані:	Значення параметру succession threshold = 1, dependency threshold = 1.00
Схема проведення тесту:	Визначити параметр succession threshold = 1, dependency threshold = 1.00. Натиснути кнопку Generate Petri net
Очікуваний результат:	Візуалізація у головному вікні програмного продукту мережі Петрі, що складається з двох місць та одного переходу, послідовно з'єднаних між собою
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі та вікно перегляду журналу подій

Таблиця 5.10 – Виявлення дискретно подійного-процесу 3

Мета тесту:	**Перевірка функції «Виявлення дискретно подійного-процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій на другому кроці
Вхідні дані:	Значення параметру succession threshold = 1, dependency threshold = 0.5, window size = 4.
Схема проведення тесту:	Визначити параметр succession threshold = 1, dependency threshold = 0.5, window size = 4. Натиснути кнопку Generate Petri net

Продовження таблиці 5.10

Очікуваний результат:	Візуалізація у головному вікні програмного продукту мережі Петрі
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі та вікно перегляду журналу подій

Таблиця 5.11 – Аналіз дискретно подійного-процесу

Мета тесту:	**Перевірка функції «Аналіз дискретно подійного-процесу»**
Початковий стан ПП	Відкрите головне вікно з мережею Петрі та вікно перегляду журналу подій.
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку Generate Report
Очікуваний результат:	Відкриття вікна звіту з аналізу дискретно-подійного процесу
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі, вікно перегляду журналу подій та вікно звіту

Таблиця 5.12 – Перегляд результатів аналізу

Мета тесту:	**Перевірка функції «Перегляд результатів аналізу»**
Початковий стан ПП	Відкрите головне вікно з мережею Петрі, вікно перегляду журналу подій та вікно звіту.
Вхідні дані:	
Схема проведення тесту:	У вікні звіту з аналізу почергово відкрити кожну з чотирьох представлених вкладок (розділів) звіту
Очікуваний результат:	Кожна вкладка (розділ) звіту містить заповнену таблицю з результатами аналізу
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі, вікно перегляду журналу подій та вікно звіту

5.3 Перевірка реалізації методу розв'язання

Функціонування розроблених та реалізованих алгоритмів перевіряється за домогою спеціально згенерованих вхідних даних – записів журналу подій. Таким чином, вхідні дані містять чітко визначений набір подій, з зазначеними частотою настання події, їх послідовністю, часовими мітками їх початку та кінця.

Вхідні дані було згенеровано засобами Microsoft Office Excel. Для цього спочатку було створено перелік усіх присутніх в процесу назв подій та задано їх середню тривалість у секундах (рисунок 5.9).

Activity 1.1	3
Activity 2.1	7
Activity 3.1	10
Activity 4.1	5
Activity 5.1	21
Activity 5.2	5
Activity 10.1	4
Activity 6.1	17
Activity 6.2	11
Activity 7.1	3
Activity 7.2	2
Activity 8.1	10
Activity 8.2	8
Activity 8.3	7
Activity 9.1	23
Activity 9.2	8
Activity 2.2	38

Рисунок 5.9 – Назви подій та їх середня тривалість

Після цього задаємо відсоток, в рамках якого допустиме відхилення значень тривалості подій від їх середнього значення тривалості (рисунок 5.10).

activity dur	
+- %	50

Рисунок 5.10 – Відхилення тривалості подій від середнього значення

Для імітації реального процесу, додамо часову затримку між завершенням однієї події та початком другої. Для цього встановимо середнє значення даної часової затримки та відсоток, в рамках якого допустиме

відхилення значень тривалості затримки від її середнього значення (рисунок 5.11).

delay dur	4
+- %	80

Рисунок 5.11 – Затримка між двома подіями

Аналогічним чином додамо затримку між кінцем однієї послідовності подій та початком другої (рисунок 5.12).

case dur	100
+- %	80

Рисунок 5.12 – Затримка між двома послідовностями подій

Після цього залишається лише дотримуючись заданих параметрів та певного набору послідовносі подій згенерувати журнал подій, що містить 2000 записів (фрагмент наведено на рисунку 5.13).

case id	activity title	start time	finish time
172	Activity 1.1	12.01.2019 09:04:25	12.01.2019 09:04:29
302	Activity 1.1	12.01.2019 09:04:26	12.01.2019 09:04:29
22	Activity 1.1	12.01.2019 09:04:27	12.01.2019 09:04:29
84	Activity 1.1	12.01.2019 09:04:27	12.01.2019 09:04:31
280	Activity 1.1	12.01.2019 09:04:27	12.01.2019 09:04:29
114	Activity 1.1	12.01.2019 09:04:30	12.01.2019 09:04:34
143	Activity 1.1	12.01.2019 09:04:30	12.01.2019 09:04:34
23	Activity 1.1	12.01.2019 09:04:31	12.01.2019 09:04:34
26	Activity 1.1	12.01.2019 09:04:31	12.01.2019 09:04:35
102	Activity 1.1	12.01.2019 09:04:31	12.01.2019 09:04:35
22	Activity 2.1	12.01.2019 09:04:32	12.01.2019 09:04:37
192	Activity 1.1	12.01.2019 09:04:33	12.01.2019 09:04:36
262	Activity 1.1	12.01.2019 09:04:33	12.01.2019 09:04:35
84	Activity 2.1	12.01.2019 09:04:34	12.01.2019 09:04:40
178	Activity 1.1	12.01.2019 09:04:34	12.01.2019 09:04:37
302	Activity 2.1	12.01.2019 09:04:34	12.01.2019 09:04:40
280	Activity 2.1	12.01.2019 09:04:35	12.01.2019 09:04:39
28	Activity 1.1	12.01.2019 09:04:36	12.01.2019 09:04:38
104	Activity 1.1	12.01.2019 09:04:36	12.01.2019 09:04:38
172	Activity 3.1	12.01.2019 09:04:36	12.01.2019 09:04:46
102	Activity 2.1	12.01.2019 09:04:37	12.01.2019 09:04:43
108	Activity 1.1	12.01.2019 09:04:37	12.01.2019 09:04:39
114	Activity 2.1	12.01.2019 09:04:37	12.01.2019 09:04:47
143	Activity 2.1	12.01.2019 09:04:37	12.01.2019 09:04:41
297	Activity 1.1	12.01.2019 09:04:37	12.01.2019 09:04:40
26	Activity 2.1	12.01.2019 09:04:38	12.01.2019 09:04:41
192	Activity 2.1	12.01.2019 09:04:38	12.01.2019 09:04:47

Рисунок 5.13 – Фрагмент згенерованого журналу подій

Зберігаємо дані в файл розширення *.csv.

Тепер, маючи знегеровані за чітко сформульованими параметрами вхідні дані, залишається подати їх на вхід розробленому програмному продукту та звіритися з результатами їх опрацювання.

Завантажуємо вхідні дані, вказуємо стовпчики ідентифікатора події, діяльності, міток часу.

Перевірямо, чи всі події та їх послідовності було виявлено програмним продуктом. Для цього скористаємося рисунками 5.9, 5.13 та 5.14.

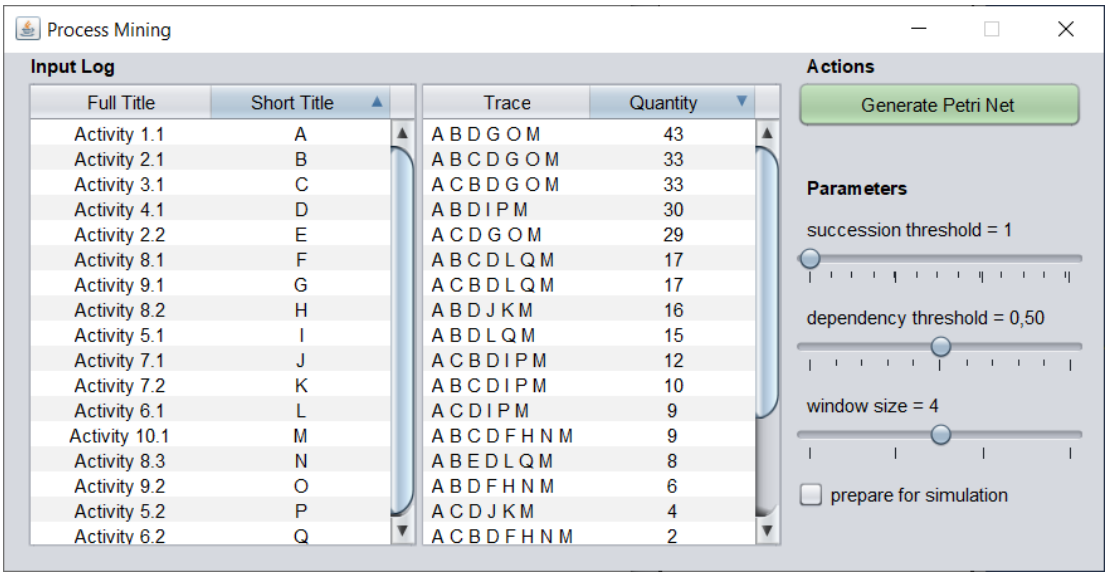


Рисунок 5.14 – Виявлені події та їх послідовності

Дійсно, рисунок 5.14 підтверджує, що програмний продукт цілком вірно виявив усі представлені події на рисунку 5.9 та усі їх можливі послідовності з файлу вхідних даних (фрагмент якого представлено на рисунку 5.13).

Генеруємо мережу Петрі (рисунок 5.15). Тепер повертаємося до послідовності подій з вхідного файлу та з рисунку 5.14 і поверевіряємо, чи знегерована мережа Петрі підтримує здійснення кожної з послідовності. Наприклад, дійсно, з події A можна перейти до B, згодом до D, G, O і M.

Таким чином робимо висновок, що дійсно результуюча мережа Петрі вірно відтворює усі послідності подій виявленого процесу.

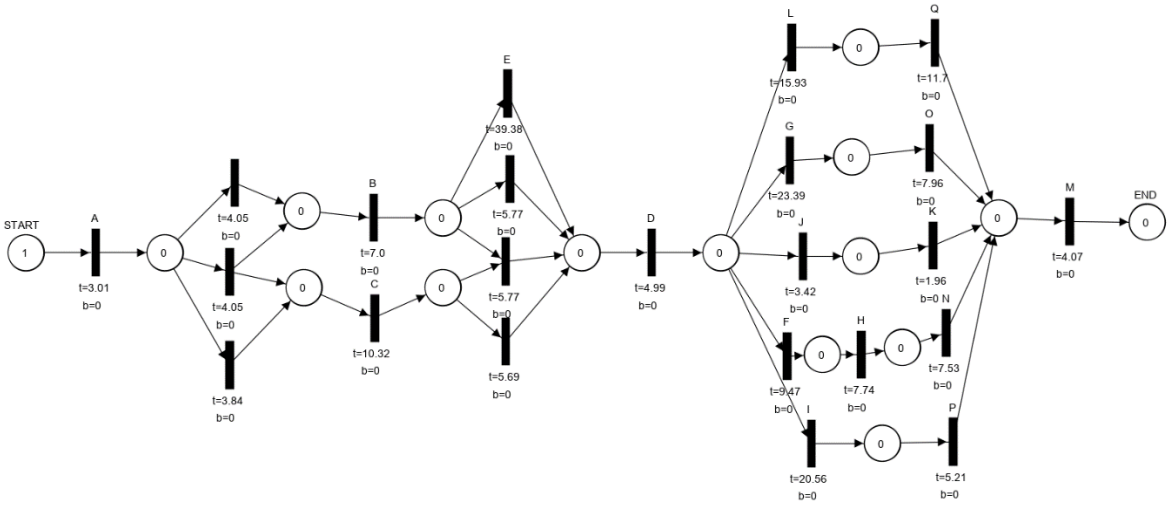
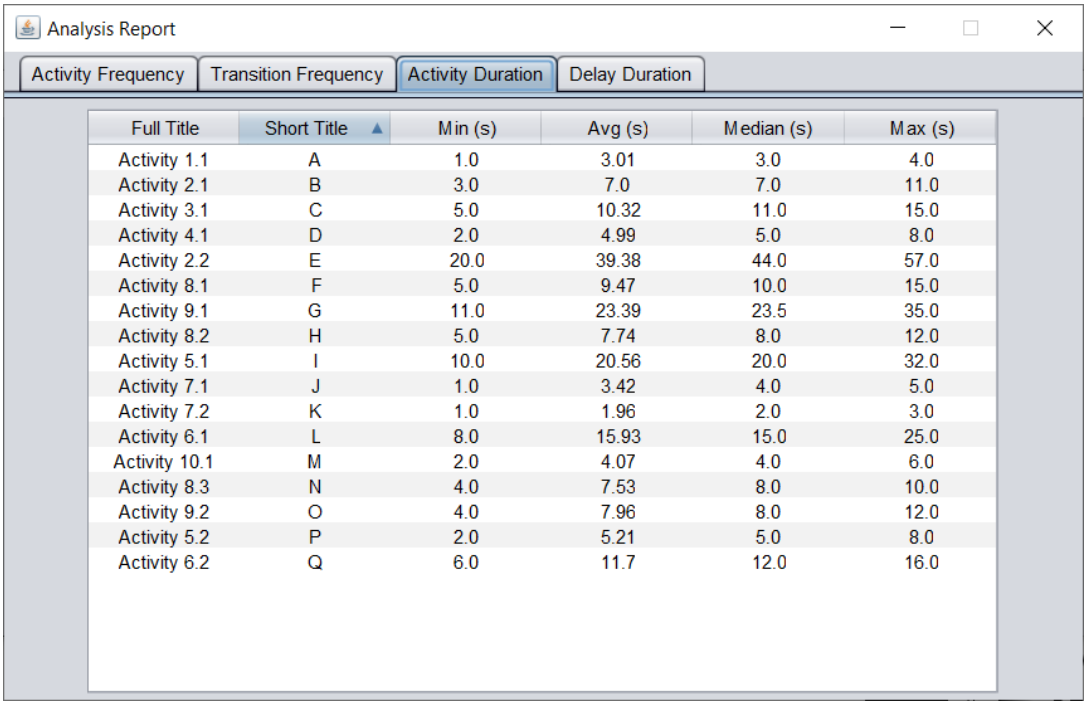


Рисунок 5.15 – Згенерована мережа Петрі

Тепер згенеруємо звіт з аналізу виявленого процесу (рисунок 5.16), та перевіримо, чи дійсно числові значення тривалості подій і затримок у вхідних даних відповідають відповідним значенням в мережі Петрі та звіті з аналізу.

Наприклад, у вхідному файлі тривалість події A (Activity 1.1) та B (Activity 2.1) становлять відповідно 3 та 7 секунд. Переглядаємо рисунки 5.15-5.16 та робимо висновок, що дійсно програмний продукт виявив середнє значення тривалості даних подій як 3.01 та 7.0 секунд відповідно.



Full Title	Short Title	Min (s)	Avg (s)	Median (s)	Max (s)
Activity 1.1	A	1.0	3.01	3.0	4.0
Activity 2.1	B	3.0	7.0	7.0	11.0
Activity 3.1	C	5.0	10.32	11.0	15.0
Activity 4.1	D	2.0	4.99	5.0	8.0
Activity 2.2	E	20.0	39.38	44.0	57.0
Activity 8.1	F	5.0	9.47	10.0	15.0
Activity 9.1	G	11.0	23.39	23.5	35.0
Activity 8.2	H	5.0	7.74	8.0	12.0
Activity 5.1	I	10.0	20.56	20.0	32.0
Activity 7.1	J	1.0	3.42	4.0	5.0
Activity 7.2	K	1.0	1.96	2.0	3.0
Activity 6.1	L	8.0	15.93	15.0	25.0
Activity 10.1	M	2.0	4.07	4.0	6.0
Activity 8.3	N	4.0	7.53	8.0	10.0
Activity 9.2	O	4.0	7.96	8.0	12.0
Activity 5.2	P	2.0	5.21	5.0	8.0
Activity 6.2	Q	6.0	11.7	12.0	16.0

Рисунок 5.16 – Звіт з аналізу тривалості подій

Аналогічним чином виконавши перевірку інших розділів звіту (частота подій, частота переходів між подіями, тривалість затримок) можна зробити висновок про безпомилкове опрацювання та аналіз вхідних даних за допомогою розробленого програмного продукту.

Окрім цього, щоб впевнитися у правильності роботи програмного продукту, проведемо такий дослід. Тестові дані, що складають 2000 записів журналу подій, перемішаємо випадковим чином, та розділимо на два журнали подій по 1000 записів. Після цього, побудоємо мережу Петрі для кожного з журналів подій та повірняємо їх. Очевидно, що дані про один й той самий процес мають давати одну й ту саму модель процесу, принаймні в загальному уявленні.

На рисунках 5.17 і 5.18 предствлено мережі Петрі, отримані з першого та другого новоутворених журналів подій.

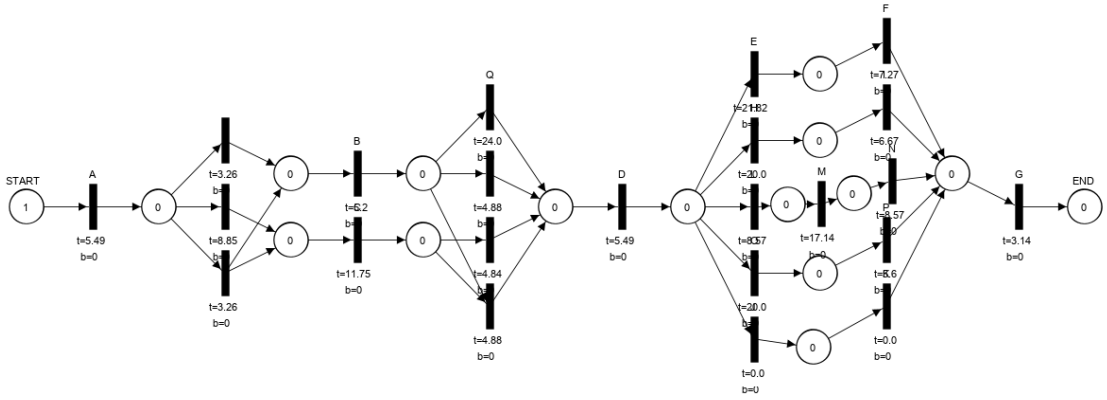


Рисунок 5.17 – Мережа Петрі з першого журналу подій

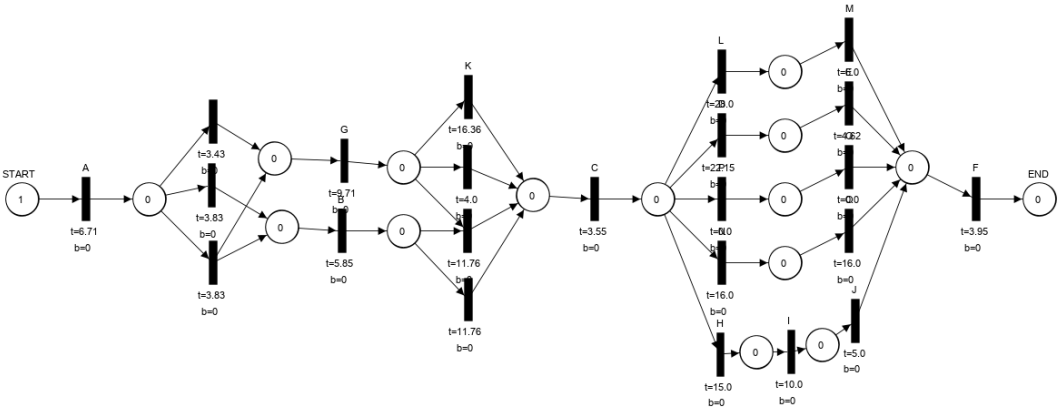


Рисунок 5.18 – Мережа Петрі з другого журналу подій

Бачимо, що попри те, що скорочені позначення подій змінилися (це пов’язано з тим, що скорочені назви присвоюються подіям в порядку їх зустрічі при аналізі в записах журналу подій), логіка та модель виявленого процесу залишилася незмінною. До того ж, виявлені мережі Петрі повністю відповідають мережі Петрі, виявленої з усіх 2000 записів журналу подій (рисунок 5.15). Це є свідченням коректної роботи розробленого програмного продукту та його відповідності поставленим вимогам на етапі проектування.

Висновок до розділу

У даному розділі було сформульовано покрокове керівництво користувача з детальним описом прикладу використання програмного продукту. Було визначено та сформульовано мету та загальні положення випробувань розробленого програмного продукту для перевірки його відповідності технічному завданню. Опис та хід тестування з зазначенням очікуваного результату та станом програмного продукту після проведення випробувань було наведено в табличному вигляді. Перевірено коректність реалізації розробленого алгоритму.

ЗАГАЛЬНІ ВИСНОВКИ

У ході виконання дипломного проекту було досліджено реальне положення справ у сфері моделювання бізнес-процесів. Особливу увагу було відведено таким областям як процесна аналітика (англ. Process Mining) та імітаційне моделювання (англ. Simulation Modeling). Було сформульовано доцільність використання методів процесної аналітики для виявлення дискретно-подійного процесу та його аналізу і подальшого його дослідження методами імітаційного моделювання.

Тож був проведений ґрунтовний аналіз предметного середовища та існуючих рішень, на основі чого було підтверджено доцільність розробки програмного продукту для аналізу та моделювання дискретно-подійного процесу на основі журналу подій.

Було сформульовано математичну постановку задачі, проаналізовано існуючі методи розв’язку поставлених задач та прийнято рішення про використання евристичного алгоритму процесної аналітики (англ. Heuristics Miner) для виявлення дискретно-подійного процесу та побудові на основі цих даних мережі Петрі.

Реалізовано евристичний алгоритм виявлення процесу та його модифікацію для отримання моделі процесу у вигляді мережі Петрі та її аналізу для надання оцінок частоти, часу тривалості та затримок між подіями.

Програмна реалізація було здійснена мовою програмування Java із застосуванням бібліотеки Discrete Event Simulation System. Було надано опис архітектури програмного забезпечення у вигляді діаграм класів, діяльності та специфікації функцій. тестування

Наведено детальне керівництво користувача та методику випробувань розробленого програмного продукту із зазначеннях ходу проведення кожного з випробувань та очікуваного результату.

Результатом роботи став програмний продукт, що на основі журналу подій:

- виявляє дискретно-подійний процес;
- будує мережу Петрі на основі виявленого процесу;
    а) виявляє події та їх послідовності;
    б) будує зв’язки розгалуження та з’єднання (як послідовного, так і паралельного виконання);
    в) при можливості спрощує модель мережі, не впливаючи на логіку її функціонування;
    г) задає часові оцінки тривалості переходів;
- аналізує виявлений процес;
    а) визначає частоту виявлених подій та переходів між ними;
    б) визначає тривалість виявлених подій та затримок між ними.

До перспектив розвитку розробленого програмного продукту можна віднести:

- повна підготовка виявленого процесу для імітаційного моделювання;
- проведення імітаційного моделювання на основі виявленої моделі;
- поглибленний аналіз виявленого процесу;
    а) визначення найбільш/найменш частих послідовностей подій довільної довжини;
    б) пошук «вузьких місць» виявленого процесу;
    в) побудова графіків та діаграм на основі проаналізованих даних;
- графічна візуалізація анімації в реальному часу виявленого процесу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Wil van der Aalst. Process Mining. Data Science in Action. Second Edition – Springer, 2015. – 477 с.
2. Jagadeesh Chandra Bose, R.P. Process Mining in the Large: Preprocessing, Discovery, and Diagnostics – Eindhoven: Technische Universiteit Eindhoven, 2012. – 386 с.
3. Gunther, Christian W. Process Mining in Flexible Environments – Eindhoven: Technische Universiteit Eindhoven, 2009. – 373 с.
4. ProM Tools [Електронний ресурс] – Режим доступу: http://www.promtools.org/doku.php – Дата доступу: 14.01.2019.
5. Discover your processes [Електронний ресурс] – Режим доступу: https://fluxicon.com/disco/ – Дата доступу: 14.01.2019.
6. Process Discovery in the Intelligent Business Cloud [Електронний ресурс] – Режим доступу: https://www.celonis.com/intelligent-business-cloud/process-discovery/ – Дата доступу: 14.01.2019.
7. myInvenio [Електронний ресурс] – Режим доступу: https://www.my-invenio.com/ – Дата доступу: 14.01.2019.
8. Ufuk Celik, Eyüp Akçetin. Process Mining Tools Comparison [Електронний ресурс] – Режим доступу: https://www.researchgate.et/publication/330683729_Process_Mining_Tools_Comparison – Дата доступу: 24.01.2019.
9. Aruna Devi .T, Dr. Kumudavalli M.V, Dr. Sudhamani. An Informative and Comparative Study of Process Mining Tools [Електронний ресурс] – Режим доступу: https://www.ijser.org/researchpaper/An-Informative-and-Comparative-Study-of-Process-Mining-Tools.pdf – Дата доступу: 24.01.2019.
10. J. Desel, W. Reisig, and G. Rozenberg, editors. Lectures on Concurrency and Petri Nets, volume 3098 of Lecture Notes in Computer Science. – Springer, 2004. – 852 с.

11. A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros. Process Mining with the Heuristics Miner Algorithm [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/229124308_Process_Mining_with_the_Heuristics_Miner-algorithm – Дата доступу: 24.01.2019.
12. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs – IEEE Transactions on Knowledge and Data Engineering, 2004. – 234 с.
13. A.J.M.M. Weijters and J.T.S. Ribeiro. Flexible Heuristics Miner (FHM) – Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, 2011. – 33 с.
14. Alves de Medeiros, Ana Karla. Genetic Process Mining – Eindhoven: Technische Universiteit Eindhoven, 2006. – 395 с.
15. A.K. Alves de Medeiros, A.J.M.M. Weijters and W.M.P. van der Aalst. Genetic Process Mining: An Experimental Evaluation [Електронний ресурс] – Режим доступу: http://wwwis.win.tue.nl/~wvdaalst/publications/p415.pdf – Дата доступу: 18.02.2019.
16. Ana Karla A. de Medeiros, A. J. M. M. Weijters, Wil M. P. van der Aalst. Genetic Process Mining: A Basic Approach and Its Challenges – Business Process Management Workshops, 2005. – 21 с.
17. Christian W. Gunther and Wil M.P. van der Aalst. Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics [Електронний ресурс] – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.1207&rep=rep1&type=pdf – Дата доступу: 24.02.2019.
18. Asst. Prof. Esmita P. Gupta. Process Mining A Comparative Study Metrics [Електронний ресурс] – Режим доступу: https://ijarcce.com/wp-content/uploads/2014/12/IJARCCE4I-a-esmita-Process-Mining.pdf – Дата доступу: 16.02.2019.

19. Intellij IDEA [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/idea/ – Дата доступу: 18.02.2019.
20. Stetsenko Inna. Discrete Event Simulation System [Електронний ресурс] – Режим доступу: https://github.com/StetsenkoInna/PetriObjModelPaint – Дата доступу: 15.01.2019.
21. Features of Java. [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/features-of-java – Дата доступу: 26.02.2019.
22. Inna Stetsenko, Kateryna Leshchenko. Stochastic Petri nets visual programming intellectual component – Technical Sciences and Technologies, 2006. – 141 с.

Додаток А

Тексти програмного коду

Аналіз та моделювання дискретно-подійного процесу на основі журналу подій

(Найменування програми (документа))

DVD-R

(Вид носія даних)

38 арк, 172 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

```
package ProcessMining.frames;
import PetriObj.PetriNet;
import ProcessMining.*;
import ProcessMining.utils.InputData;
import graphnet.GraphPetriNet;
import graphpresentation.FileUse;
import graphpresentation.PetriNetsFrame;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.TableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
import java.util.List;
public class ProcessMiningFrame extends JFrame {
    private int FRAME_WIDTH = 700;
    private int FRAME_HEIGHT = 330;
    private int PARAMS_WIDTH = 200;
    private int COMPONENT_HEIGHT = 30;
    private int SPACING_HEIGHT = 10;
    private JPanel paramsBox;
    private JPanel actionBox;
    private JPanel logPanel;
    private JFrame frame;
    private ReportFrame reportFrame;
    private JButton reportButton;
    private InputData inputData;
    private PetriNetsFrame petriNetsFrame;
    private HeuristicMiner miner;
    private boolean isReportOpened;
    public ProcessMiningFrame(List<List<String>> table, PetriNetsFrame
petriNetsFrame){
        super();
        this.petriNetsFrame = petriNetsFrame;
        miner = new HeuristicMiner();
        init(table);
    }
    private JPanel getNextParametersComponents(){
        // get thresholds MAX values
        miner.calculateDirectSuccessionMatrix(inputData.getLog());
        int MAX_WINDOWS_SIZE = miner.getMaxTraceLength() - 1;
        int MAX_SUCCESSION_THRESHOLD = miner.getMaxDirectSuccessionValue();
        JPanel paramsBox = new JPanel();
        paramsBox.setLayout(new BoxLayout(paramsBox, BoxLayout.Y_AXIS));
        // params components
        paramsBox.add(Box.createRigidArea(new Dimension(0,
SPACING_HEIGHT*3)));
        JLabel label = new JLabel("  Parameters ", SwingConstants.LEFT);
        label.setFont(label.getFont().deriveFont(label.getFont().getStyle() |
Font.BOLD));
        paramsBox.add(label);
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        ////////////////////////////// THRESHOLD 1
        String thresholdName1 = "succession threshold";
        int thresholdMin1 = 1;
        int thresholdMax1 = MAX_SUCCESSION_THRESHOLD;
        int delta1 = thresholdMax1 - thresholdMin1;
        JLabel thresholdLabel1 = new JLabel("  " + thresholdName1 + " = " +
String.format("%d", thresholdMin1) + " ");
```

```
        paramsBox.add(thresholdLabel1);
        JSlider thresholdSlider1 = new JSlider(JSlider.HORIZONTAL,
thresholdMin1,thresholdMax1, thresholdMin1);
        int spacing = delta1/3 > 0 ? delta1/3 : 1;
        thresholdSlider1.setMajorTickSpacing(spacing);
        thresholdSlider1.setMinorTickSpacing(spacing / 4);
        thresholdSlider1.setPaintTicks(true);
        thresholdSlider1.setPreferredSize(new
Dimension(PARAMS_WIDTH/2,COMPONENT_HEIGHT));
        thresholdSlider1.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        thresholdSlider1.setAlignmentX(Component.LEFT_ALIGNMENT);
        thresholdSlider1.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                JSlider source = (JSlider)e.getSource();
                thresholdLabel1.setText("  " + thresholdName1 + " = " +
source.getValue() + " ");
            }
        });
        paramsBox.add(thresholdSlider1);
        ////////////////////////////// THRESHOLD 2
        String thresholdName2 = "dependency threshold";
        double thresholdMin2 = 0;
        double thresholdMax2 = 1;
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        JLabel thresholdLabel2 = new JLabel("  " + thresholdName2 + " = " +
String.format("%.2f", (thresholdMin2+thresholdMax2)*0.5) + " ");
        paramsBox.add(thresholdLabel2);
        JSlider thresholdSlider2 = new JSlider(JSlider.HORIZONTAL, 0,100,50);
        thresholdSlider2.setMajorTickSpacing(50);
        thresholdSlider2.setMinorTickSpacing(10);
        thresholdSlider2.setPaintTicks(true);
        thresholdSlider2.setPreferredSize(new
Dimension(PARAMS_WIDTH/2,COMPONENT_HEIGHT));
        thresholdSlider2.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        thresholdSlider2.setAlignmentX(Component.LEFT_ALIGNMENT);
        thresholdSlider2.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                JSlider source = (JSlider)e.getSource();
                String value = String.format("%.2f", source.getValue()*0.01);
                thresholdLabel2.setText("  " + thresholdName2 + " = " +
String.format("%.2f", source.getValue()*0.01) + " ");
            }
        });
        paramsBox.add(thresholdSlider2);
        ////////////////////////////// THRESHOLD 3
        String thresholdName3 = "window size";
        int thresholdMin3 = 1;
        int thresholdMax3 = MAX_WINDOWS_SIZE;
        int delta3 = thresholdMax3 - thresholdMin3;
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        JLabel thresholdLabel3 = new JLabel("  " + thresholdName3 + " = " +
String.format("%d", Math.round(thresholdMin3 + (50*0.01)*(delta3))) + " ");
        paramsBox.add(thresholdLabel3);
        JSlider thresholdSlider3 = new JSlider(JSlider.HORIZONTAL,
thresholdMin3,thresholdMax3,(int)Math.round(thresholdMin3 +
(50*0.01)*(delta3)));
        spacing = delta3/3 > 0 ? delta3/3 : 1;
        thresholdSlider3.setMajorTickSpacing(spacing);
        thresholdSlider3.setMinorTickSpacing(spacing / 4);
        thresholdSlider3.setPaintTicks(true);
```

```
        thresholdSlider3.setPreferredSize(new
Dimension(PARAMS_WIDTH/2,COMPONENT_HEIGHT));
        thresholdSlider3.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        thresholdSlider3.setAlignmentX(Component.LEFT_ALIGNMENT);
        thresholdSlider3.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                JSlider source = (JSlider)e.getSource();
                thresholdLabel3.setText("  " + thresholdName3 + " = " +
source.getValue() + " ");
            }
        });
        paramsBox.add(thresholdSlider3);
        ///
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        JCheckBox simulationCheckBox = new JCheckBox("prepare for
simulation");
        paramsBox.add(simulationCheckBox);
        JButton runButton = new JButton("Generate Petri Net ");
        runButton.setBackground(new Color(132, 169, 126));
        runButton.setMaximumSize(new
Dimension(PARAMS_WIDTH*100,COMPONENT_HEIGHT));
        runButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int t1 = thresholdSlider1.getValue();
                double t2 = thresholdSlider2.getValue()*0.01;
                int t3 = thresholdSlider3.getValue();
                inputData.setSuccessionThreshold(t1);
                inputData.setDependencyThreshold(t2);
                inputData.setWindowSize(t3);

inputData.setPrepareForSimulation(simulationCheckBox.isSelected());
                ////////////
                JScrollPane pane =
petriNetsFrame.GetPetriNetPanelScrollPane();
                Point paneCenter = new
Point(pane.getLocation().x+pane.getBounds().width/2,
pane.getLocation().y+pane.getBounds().height/2);
                FileUse fileUse = new FileUse();
                PetriNet net = miner.runConsecutively(inputData);
                if (net!= null) {
                    GraphPetriNet graphNet =
fileUse.generateGraphNetBySimpleNet(petriNetsFrame.getPetriNetsPanel(), net,
paneCenter);
                    petriNetsFrame.getPetriNetsPanel().addGraphNet(graphNet);
                    petriNetsFrame.getPetriNetsPanel().repaint();
                } else {
                    JOptionPane.showMessageDialog(frame, "Petri Net is
disjointed.\nTry to change threshold parameters", "Generation is impossible",
                            JOptionPane.WARNING_MESSAGE);
                }
                ////////
                isReportOpened = false;
                if (reportButton == null) {
                    //paramsBox.add(Box.createRigidArea(new Dimension(0,
SPACING_HEIGHT)));
                    reportButton = new JButton("Generate Report ");
                    //hideButton.setBackground(new Color(132, 169, 126));
                    reportButton.setMaximumSize(new
Dimension(PARAMS_WIDTH*100,COMPONENT_HEIGHT));
                    reportButton.addActionListener(new ActionListener() {
                        @Override
```

```
                        public void actionPerformed(ActionEvent e) {
                            if (!isReportOpened || reportFrame == null ||
reportFrame.isClosed()) {
                                reportFrame = new ReportFrame(miner);
                                isReportOpened = true;
                            }
                        }
                    });
                    paramsBox.add(reportButton,1);
                    frame.pack();
                }
            }
        });
        paramsBox.add(runButton,0);
        return paramsBox ;
    }
    private JPanel getTracesInfoComponents(InputData inputData){
        JPanel logPanel = new JPanel(new BorderLayout());
        logPanel.setPreferredSize(new Dimension(FRAME_WIDTH - PARAMS_WIDTH,
FRAME_HEIGHT));
        //logPanel.setBorder(BorderFactory.createTitledBorder(new
EmptyBorder(0, 10, 10, 0),"   Input Log "));
        DefaultTableCellRenderer centerRenderer = new
DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment( JLabel.CENTER );
        Object[] columnNames = Arrays.asList("Full Title","Short
Title").toArray();
        Object[][] data = new
Object[inputData.getActivityFullToShortTitle().size()][2];
        Iterator<String> iterator =
inputData.getActivityFullToShortTitle().keySet().iterator();
        for (int i = 0; i < inputData.getActivityFullToShortTitle().size();
i++) {
            String key = iterator.next();
            String value = inputData.getActivityFullToShortTitle().get(key);
            data[i] = Arrays.asList(key, value).toArray();
        }
        JTable table = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
        };
        final TableRowSorter<TableModel> sorter = new
TableRowSorter<TableModel>(table.getModel());
        table.setRowSorter(sorter);
        table.getRowSorter().toggleSortOrder(1);
        table.setDefaultRenderer(Object.class, centerRenderer);

((DefaultTableCellRenderer)table.getTableHeader().getDefaultRenderer())
                .setHorizontalAlignment(JLabel.CENTER);
        JScrollPane tableScrollPane = new JScrollPane(table);
        tableScrollPane.setPreferredSize(new Dimension((FRAME_WIDTH -
PARAMS_WIDTH)/2, FRAME_HEIGHT));
        table.setFillsViewportHeight(true);
        logPanel.add(tableScrollPane, BorderLayout.WEST);
        columnNames = Arrays.asList("Trace", "Quantity").toArray();
        data = new Object[inputData.getLog().getLogItems().size()][2];
        for (int i = 0; i < inputData.getLog().getLogItems().size(); i++) {
            LogItem li = inputData.getLog().getLogItems().get(i);
            String trace = "";
            for (int j = 0; j < li.getTrace().size(); j++) {
                trace += li.getTrace().get(j) + " ";
            }
            data[i] = Arrays.asList(trace, li.getFrequency()).toArray();
```

```
        }
        JTable table2 = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
        };
        final TableRowSorter<TableModel> sorter2 = new
TableRowSorter<TableModel>(table2.getModel());
        sorter2.setComparator(1, new Comparator<Integer>() {
            @Override
            public int compare(Integer o1, Integer o2) {
                return Integer.compare(o1, o2);
            }
        });
        table2.setRowSorter(sorter2);
        table2.getRowSorter().toggleSortOrder(1);
        table2.getRowSorter().toggleSortOrder(1);
        table2.getColumnModel().getColumn(1).setCellRenderer( centerRenderer
);

((DefaultTableCellRenderer)table2.getTableHeader().getDefaultRenderer())
                .setHorizontalAlignment(JLabel.CENTER);
        JScrollPane tableScrollPane2 = new JScrollPane(table2);
        tableScrollPane2.setPreferredSize(new Dimension((FRAME_WIDTH -
PARAMS_WIDTH)/2, FRAME_HEIGHT));
        table2.setFillsViewportHeight(true);
        logPanel.add(tableScrollPane2, BorderLayout.CENTER);
        return logPanel;
    }
    private void init(List<List<String>> list){
        frame = new JFrame("Process Mining");
        frame.setLayout(new BorderLayout());
        logPanel = new JPanel(new BorderLayout());
        logPanel.setPreferredSize(new Dimension(FRAME_WIDTH - PARAMS_WIDTH,
FRAME_HEIGHT));
        logPanel.setBorder(BorderFactory.createTitledBorder(new
EmptyBorder(0, 10, 10, 0),"   Input Log "));
        Object[] columnNames = list.get(0).toArray();
        Object[][] data = new Object[list.size()-1][list.get(0).size()];
        for (int i = 1; i < list.size(); i++) {
            data[i-1] = list.get(i).toArray();
        }
        JTable table = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
        };
        final TableRowSorter<TableModel> sorter = new
TableRowSorter<TableModel>(table.getModel());
        table.setRowSorter(sorter);
        JScrollPane tableScrollPane = new JScrollPane(table);
        table.setFillsViewportHeight(true);
        logPanel.add(tableScrollPane);
        JPanel paramsPanel = new JPanel(new BorderLayout());
        paramsPanel.setSize(PARAMS_WIDTH, FRAME_HEIGHT);
        paramsPanel.setPreferredSize(new Dimension(PARAMS_WIDTH,
FRAME_HEIGHT));
        paramsPanel.setBorder(BorderFactory.createTitledBorder(new
EmptyBorder(0, 0, 10, 10)," Actions"));
        paramsBox = new JPanel();
        paramsBox.setLayout(new BoxLayout(paramsBox, BoxLayout.Y_AXIS));
        int COMPONENT_HEIGHT = 30;
        int SPACING_HEIGHT = 10;
        // params components
```

```
        paramsBox.add(Box.createRigidArea(new Dimension(0,
SPACING_HEIGHT*3)));
        JLabel label = new JLabel("  Parameters ", SwingConstants.LEFT);
        label.setFont(label.getFont().deriveFont(label.getFont().getStyle() |
Font.BOLD));
        paramsBox.add(label);
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        JLabel label1 = new JLabel("  case id column ", SwingConstants.LEFT);
        paramsBox.add(label1);
        JComboBox eventIdComboBox = new JComboBox(columnNames);
        eventIdComboBox.setAlignmentX(Component.LEFT_ALIGNMENT);
        eventIdComboBox.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        eventIdComboBox.setPrototypeDisplayValue("XXXXXX");
        eventIdComboBox.setSelectedIndex(0);
        paramsBox.add(eventIdComboBox);
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        paramsBox.add(new JLabel("  activity column "));
        JComboBox activityComboBox = new JComboBox(columnNames);
        activityComboBox.setAlignmentX(Component.LEFT_ALIGNMENT);
        activityComboBox.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        activityComboBox.setPrototypeDisplayValue("XXXXXX");
        activityComboBox.setSelectedIndex(1);
        paramsBox.add(activityComboBox);
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        paramsBox.add(new JLabel("  start timestamp column "));
        JComboBox timeComboBox = new JComboBox(columnNames);
        timeComboBox.setAlignmentX(Component.LEFT_ALIGNMENT);
        timeComboBox.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        timeComboBox.setPrototypeDisplayValue("XXXXXX");
        timeComboBox.setSelectedIndex(2);
        paramsBox.add(timeComboBox);
        paramsBox.add(Box.createRigidArea(new Dimension(0, SPACING_HEIGHT)));
        paramsBox.add(new JLabel("  finish timestamp column "));
        JComboBox timeComboBox2 = new JComboBox(columnNames);
        timeComboBox2.setAlignmentX(Component.LEFT_ALIGNMENT);
        timeComboBox2.setMaximumSize(new
Dimension(PARAMS_WIDTH*10,COMPONENT_HEIGHT));
        timeComboBox2.setPrototypeDisplayValue("XXXXXX");
        timeComboBox2.setSelectedIndex(3);
        paramsBox.add(timeComboBox2);
        JButton nextButton = new JButton("Next Step ");
        nextButton.setBackground(new Color(132, 169, 126));
        nextButton.setMaximumSize(new
Dimension(PARAMS_WIDTH*100,COMPONENT_HEIGHT));
        nextButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                inputData = new InputData();
                int id1 = eventIdComboBox.getSelectedIndex();
                int id2 = activityComboBox.getSelectedIndex();
                int id3 = timeComboBox.getSelectedIndex();
                int id4 = timeComboBox2.getSelectedIndex();
                List<List<String>> columns = new ArrayList<>();
                columns.add(new ArrayList<>());
                columns.add(new ArrayList<>());
                columns.add(new ArrayList<>());
                columns.add(new ArrayList<>());
                for (int i = 1; i < list.size(); i++) {
                    columns.get(0).add(list.get(i).get(id1));
                    columns.get(1).add(list.get(i).get(id2));
                    columns.get(2).add(list.get(i).get(id3));
```

```
                    columns.get(3).add(list.get(i).get(id4));
                }
                inputData.setCaseIdColumn(columns.get(0));
                inputData.setActivityColumn(columns.get(1));
                inputData.setBeginTimestampColumn(columns.get(2));
                inputData.setEndTimestampColumn(columns.get(3));
                inputData.initActivityFullToShortTitle();
                String error = inputData.initLog();
                if (error.equals("")) {
                    paramsBox.removeAll();
                    paramsBox.add(getNextParametersComponents());
                    logPanel.removeAll();
                    logPanel.add(getTracesInfoComponents(inputData));
                    frame.pack();
                    frame.repaint();
                } else {
                    JOptionPane.showMessageDialog(frame, error, "Oops,
error!",
                            JOptionPane.WARNING_MESSAGE);
                }
            }
        });
        paramsBox.add(nextButton,0);
        JScrollPane paramsScrollPane = new JScrollPane(paramsBox);
        paramsScrollPane.setBorder(new EmptyBorder(0, 0, 0, 0));
        paramsPanel.add(paramsScrollPane);
        frame.add(logPanel, BorderLayout.CENTER);
        frame.add(paramsPanel, BorderLayout.EAST);
        frame.setVisible(true);
        //frame.setSize(FRAME_WIDTH,FRAME_HEIGHT);
        frame.pack();
        //frame.setMinimumSize(new Dimension(PARAMS_WIDTH+80,
FRAME_HEIGHT+120));
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
    }
}
package ProcessMining.frames;
import ProcessMining.HeuristicMiner;
import ProcessMining.utils.InputData;
import ProcessMining.utils.IntPair;
import javax.swing.*;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.TableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.*;
public class ReportFrame extends JFrame{
    private int FRAME_WIDTH = 700;
    private int FRAME_HEIGHT = 450;
    private int SPACING_HEIGHT = 10;
    private boolean isClosed;
    public ReportFrame(HeuristicMiner miner){
        super("Analysis Report");
        init(miner);
    }
    public double round(double value, int places) {
        if (places < 0) throw new IllegalArgumentException();
        BigDecimal bd = new BigDecimal(value);
        bd = bd.setScale(places, RoundingMode.HALF_UP);
```

```
        return bd.doubleValue();
    }
    private void init(HeuristicMiner miner){
        setLayout(new BorderLayout());
        setSize(new Dimension(FRAME_WIDTH, FRAME_HEIGHT));
        setPreferredSize(new Dimension(FRAME_WIDTH, FRAME_HEIGHT));
        JTabbedPane tabbedPane = new JTabbedPane();
        JPanel panel0 = new JPanel();
        JPanel panel1 = new JPanel();
        JPanel panel2 = new JPanel();
        JPanel panel3 = new JPanel();
        JPanel panel4 = new JPanel();
        //tabbedPane.addTab("Overall Statistics", panel0);
        tabbedPane.addTab("Activity Frequency", panel1);
        tabbedPane.addTab("Transition Frequency", panel2);
        tabbedPane.addTab("Activity Duration", panel3);
        tabbedPane.addTab("Delay Duration", panel4);
        DefaultTableCellRenderer centerRenderer = new
DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment( JLabel.CENTER );
        HashMap<String, Integer> activityFrequency =
miner.getActivityFrequency();
        HashMap<IntPair, Integer> transitionFrequency =
miner.getTransitionFrequency();
        HashMap<String, ArrayList<Long>> activityDuration =
miner.getActivityDuration();
        HashMap<IntPair, ArrayList<Long>> delayDuration =
miner.getDelayDuration();
        InputData inputData = miner.getInputData();
        ////////////////////////////////////
        Object[] columnNames = Arrays.asList("Full Title","Short Title",
"Frequency", "Frequency (%)").toArray();
        Object[][] data = new Object[activityFrequency.keySet().size()][4];
        int freqSum = 0;
        for (int e: activityFrequency.values()) {
            freqSum += e;
        }
        Iterator<String> iterator =
inputData.getActivityFullToShortTitle().keySet().iterator();
        for (int i = 0; i < inputData.getActivityFullToShortTitle().size();
i++) {
            String key = iterator.next();
            String value = inputData.getActivityFullToShortTitle().get(key);
            double percent = 0;
            if (freqSum > 0){
                percent = activityFrequency.get(value)*100d/freqSum;
            }
            data[i] = Arrays.asList(key, value, activityFrequency.get(value),
round(percent,2)).toArray();
        }
        JTable table = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
        };
        final TableRowSorter<TableModel> sorter = new
TableRowSorter<TableModel>(table.getModel());
        sorter.setComparator(3, new Comparator<Double>() {
            @Override
            public int compare(Double o1, Double o2) {
                return Double.compare(o1, o2);
            }
        });
        sorter.setComparator(2, new Comparator<Integer>() {
```

```
            @Override
            public int compare(Integer o1, Integer o2) {
                return Integer.compare(o1, o2);
            }
        });
        table.setRowSorter(sorter);
        table.getRowSorter().toggleSortOrder(2);
        table.getRowSorter().toggleSortOrder(2);
        table.setDefaultRenderer(Object.class, centerRenderer);

((DefaultTableCellRenderer)table.getTableHeader().getDefaultRenderer())
                .setHorizontalAlignment(JLabel.CENTER);
        JScrollPane tableScrollPane = new JScrollPane(table);
        tableScrollPane.setAlignmentX(Component.CENTER_ALIGNMENT);
        tableScrollPane.setPreferredSize(new Dimension(FRAME_WIDTH*5/6,
FRAME_HEIGHT *5/6));
        table.setFillsViewportHeight(true);
        panel1.add(tableScrollPane);
        ////////////////////////////////////
        columnNames = Arrays.asList("From, Full Title","From, Short Title",
"To, Full Title","To, Short Title", "Frequency","Frequency (%)").toArray();
        ArrayList<Object> dataList = new ArrayList<>();
        Iterator<IntPair> iterator2 =
transitionFrequency.keySet().iterator();
        freqSum = 0;
        for (int e: transitionFrequency.values()) {
            freqSum += e;
        }
        while (iterator2.hasNext()) {
            IntPair key = iterator2.next();
            int value = transitionFrequency.get(key);
            if  (value > 0) {
                String from =
inputData.getLog().getActivityNumberToTitleMap().get(key.x);
                String to =
inputData.getLog().getActivityNumberToTitleMap().get(key.y);
                String fromFull =
inputData.getActivityShortToFullTitle().get(from);
                String toFull =
inputData.getActivityShortToFullTitle().get(to);
                double percent = 0;
                if (freqSum > 0){
                    percent = transitionFrequency.get(key)*100d/freqSum;
                }
                dataList.add(Arrays.asList(fromFull, from,toFull,to, value,
round(percent,2)).toArray());
            }
        }
        data = new Object[dataList.size()][6];
        for (int i = 0; i < dataList.size(); i++) {
            data[i] = (Object[])dataList.get(i);
        }
        JTable table2 = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
        };
        final TableRowSorter<TableModel> sorter2 = new
TableRowSorter<TableModel>(table2.getModel());
        sorter2.setComparator(5, new Comparator<Double>() {
            @Override
            public int compare(Double o1, Double o2) {
                return Double.compare(o1, o2);
            }
```

```
        });
        sorter2.setComparator(4, new Comparator<Integer>() {
            @Override
            public int compare(Integer o1, Integer o2) {
                return Integer.compare(o1, o2);
            }
        });
        table2.setRowSorter(sorter2);
        table2.getRowSorter().toggleSortOrder(4);
        table2.getRowSorter().toggleSortOrder(4);
        table2.setDefaultRenderer(Object.class, centerRenderer);

((DefaultTableCellRenderer)table2.getTableHeader().getDefaultRenderer())
                .setHorizontalAlignment(JLabel.CENTER);
        JScrollPane tableScrollPane2 = new JScrollPane(table2);
        tableScrollPane2.setAlignmentX(Component.CENTER_ALIGNMENT);
        tableScrollPane2.setPreferredSize(new Dimension(FRAME_WIDTH*5/6,
FRAME_HEIGHT *5/6));
        table2.setFillsViewportHeight(true);
        panel2.add(tableScrollPane2);
        ////////////////////////////////////
        String timeUnit = " (s)";
        columnNames = Arrays.asList("Full Title","Short Title",
"Min"+timeUnit, "Avg"+timeUnit, "Median"+timeUnit, "Max"+timeUnit).toArray();
        data = new Object[activityFrequency.keySet().size()][6];
        iterator =
inputData.getActivityFullToShortTitle().keySet().iterator();
        for (int i = 0; i < inputData.getActivityFullToShortTitle().size();
i++) {
            String key = iterator.next();
            String value = inputData.getActivityFullToShortTitle().get(key);
            ArrayList<Long> list = activityDuration.get(value);
            Collections.sort(list);
            double min = Collections.min(list);
            double max = Collections.max(list);
            double sum = 0;
            double avg = 0;
            if (list.size() > 0) {
                for (Long l: list) {
                    sum += l;
                }
                avg = sum*1f/list.size();
            }
            double med = 0;
            if (list.size()%2==0){
                Long next = list.size() > 2 ? list.get(list.size()/2+1) :
list.get(list.size()/2);
                med = (list.get(list.size()/2)+next)*0.5;
            } else {
                med = list.get(list.size()/2);
            }
            data[i] = Arrays.asList(key, value, round(min,2), round(avg,2),
round(med,2),round(max,2)).toArray();
        }
        JTable table3 = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
            public Class getColumnClass(int column) {
                Class returnValue;
                if ((column >= 0) && (column < getColumnCount())) {
                    returnValue = getValueAt(0, column).getClass();
                } else {
                    returnValue = Object.class;
```

```
                }
                return returnValue;
            }
        };
        final TableRowSorter<TableModel> sorter3 = new
TableRowSorter<TableModel>(table3.getModel());
        for (int i = 2; i <= 5; i++) {
            sorter3.setComparator(i, new Comparator<Double>() {
                @Override
                public int compare(Double o1, Double o2) {
                    return Double.compare(o1, o2);
                }
            });
        }
        table3.setRowSorter(sorter3);
        table3.getRowSorter().toggleSortOrder(3);
        table3.getRowSorter().toggleSortOrder(3);
        table3.setDefaultRenderer(Object.class, centerRenderer);

((DefaultTableCellRenderer)table3.getTableHeader().getDefaultRenderer())
                .setHorizontalAlignment(JLabel.CENTER);
        JScrollPane tableScrollPane3 = new JScrollPane(table3);
        tableScrollPane3.setAlignmentX(Component.CENTER_ALIGNMENT);
        tableScrollPane3.setPreferredSize(new Dimension(FRAME_WIDTH*5/6,
FRAME_HEIGHT *5/6));
        table3.setFillsViewportHeight(true);
        panel3.add(tableScrollPane3);
        ////////////////////////////////////
        columnNames = Arrays.asList("From, Full Title","From, Short
Title","To, Full Title","To, Short Title", "Min"+timeUnit, "Avg"+timeUnit,
"Median"+timeUnit, "Max"+timeUnit).toArray();
        data = new Object[delayDuration.size()][8];
        Iterator<IntPair> iterator3 = delayDuration.keySet().iterator();
        for (int i = 0; i < delayDuration.size(); i++) {
            IntPair key = iterator3.next();
            String value = inputData.getActivityFullToShortTitle().get(key);
            ArrayList<Long> list = delayDuration.get(key);
            Collections.sort(list);
            double min = Collections.min(list);
            double max = Collections.max(list);
            double sum = 0;
            double avg = 0;
            if (list.size() > 0) {
                for (Long l: list) {
                    sum += l;
                }
                avg = sum*1f/list.size();
            }
            double med = 0;
            if (list.size()%2==0){
                Long next = list.size() > 2 ? list.get(list.size()/2+1) :
list.get(list.size()/2);
                med = (list.get(list.size()/2)+next)*0.5;
            } else {
                med = list.get(list.size()/2);
            }
            String from =
inputData.getLog().getActivityNumberToTitleMap().get(key.x);
            String to =
inputData.getLog().getActivityNumberToTitleMap().get(key.y);
            String fromFull =
inputData.getActivityShortToFullTitle().get(from);
            String toFull = inputData.getActivityShortToFullTitle().get(to);
            data[i] = Arrays.asList(fromFull, from,toFull,to, round(min,2),
```

```
round(avg,2), round(med,2),round(max,2)).toArray();
        }
        JTable table4 = new JTable(data, columnNames){
            public boolean isCellEditable(int row, int col){
                return false;
            }
        };
        final TableRowSorter<TableModel> sorter4 = new
TableRowSorter<TableModel>(table4.getModel());
        for (int i = 4; i <= 7; i++) {
            sorter4.setComparator(i, new Comparator<Double>() {
                @Override
                public int compare(Double o1, Double o2) {
                    return Double.compare(o1, o2);
                }
            });
        }
        table4.setRowSorter(sorter4);
        table4.getRowSorter().toggleSortOrder(5);
        table4.getRowSorter().toggleSortOrder(5);
        table4.setDefaultRenderer(Object.class, centerRenderer);

((DefaultTableCellRenderer)table4.getTableHeader().getDefaultRenderer())
                .setHorizontalAlignment(JLabel.CENTER);
        JScrollPane tableScrollPane4 = new JScrollPane(table4);
        tableScrollPane4.setAlignmentX(Component.CENTER_ALIGNMENT);
        tableScrollPane4.setPreferredSize(new Dimension(FRAME_WIDTH*5/6,
FRAME_HEIGHT *5/6));
        table4.setFillsViewportHeight(true);
        panel4.add(tableScrollPane4);
        getContentPane().add(tabbedPane);
        setVisible(true);
        //pack();
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setResizable(false);
        isClosed = false;
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                isClosed = true;
            }
        });
    }
    public boolean isClosed(){
        return isClosed;
    }
}
package ProcessMining.utils;
import ProcessMining.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
public class InputData {
    private List<String> caseIdColumn;
    private List<String> activityColumn;
    private List<String> beginTimestampColumn;
    private List<String> endTimestampColumn;
    private int successionThreshold;
    private double dependencyThreshold;
    private int windowSize;
    private Log log;
```

```
    private HashMap<String,String> activityFullToShortTitle;
    private HashMap<String,String> activityShortToFullTitle;
    private boolean prepareForSimulation;
    private int characterPointer;
    private int numberPointer;
    private boolean isLogValid;
    private String startActivity;
    private String endActivity;
    private HashMap<Integer, ArrayList<Integer>> numericTraces;
    public String initLog(){
        String errorMessage = "";
        boolean isOnlyStart = true;
        boolean isOnlyFinish = true;
        String start = "";
        String finish = "";
        numericTraces = new HashMap<>();
        HashMap<Integer, String> traces = new HashMap<>();
        for (int i = 0; i < caseIdColumn.size(); i++) {
            int id = 0;
            try {
                id = Integer.parseInt(caseIdColumn.get(i));
            } catch (Exception e) {return "'case id column' is incorrect";}
            String trace = "";
            ArrayList<Integer> list = new ArrayList<>();
            if (traces.containsKey(id)) {
                trace = traces.get(id);
                list = numericTraces.get(id);
            }
            String traceItem =
activityFullToShortTitle.get(activityColumn.get(i));
            trace += traceItem + " ";
            list.add(i);
            traces.put(id, trace);
            numericTraces.put(id, list);
            if (i == 0) {
                if (start.equals("")) {
                    start = traceItem;
                } else if (!start.equals(traceItem)) {
                    isOnlyStart = false;
                }
            }
            if (i == caseIdColumn.size() - 1) {
                if (finish.equals("")) {
                    finish = traceItem;
                } else if (!finish.equals(traceItem)) {
                    isOnlyFinish = false;
                }
            }
        }
        HashMap<String, Integer> traceQuantity = new HashMap<>();
        for (String s : traces.values()) {
            int quantity = 1;
            String key = s.trim();
            if (traceQuantity.containsKey(key)) {
                quantity += traceQuantity.get(key);
            }
            traceQuantity.put(key, quantity);
        }
        log = new Log();
        for (String key : traceQuantity.keySet()) {
            log.add(new LogItem(key, traceQuantity.get(key)));
        }
        isLogValid = isOnlyFinish && isOnlyStart;
        startActivity = start;
```

```
        endActivity = finish;
        DateTimeFormatter f = DateTimeFormatter.ofPattern("dd.MM.yyyy
HH:mm:ss");
        LocalDateTime begin, end;
        for (int i = 0; i < beginTimestampColumn.size(); i++) {
            try {
                begin =
LocalDateTime.from(f.parse(beginTimestampColumn.get(i)));
            } catch (Exception e) { return "Can't read 'start timestamp
column'";}
            try {
            end = LocalDateTime.from(f.parse(endTimestampColumn.get(i)));
            } catch (Exception e) { return "Can't read 'finish timestamp
column'";}
            long dur = ChronoUnit.SECONDS.between(begin, end);
            if (dur < 0) {
                return "'start timestamp column' and 'finish timestamp\n
column' are incorrect";
            }
        }
        return errorMessage;
    }
    public void initActivityFullToShortTitle(){
        activityFullToShortTitle = new HashMap<>();
        activityShortToFullTitle = new HashMap<>();
        characterPointer = 65;
        numberPointer = 0;
        for (String s : activityColumn) {
            if (!activityFullToShortTitle.containsKey(s)) {
                String number = numberPointer > 0 ?
String.valueOf(numberPointer) : "";
                activityFullToShortTitle.put(s,
(char)characterPointer+number);
                activityShortToFullTitle.put((char)characterPointer+number,
s);
                characterPointer++;
                if (characterPointer > 90) {
                    characterPointer = 65;
                    numberPointer++;
                }
            }
        }
        /*for (String s :activityFullToShortTitle.keySet()) {
            System.out.println(s + "  -->  " +
activityFullToShortTitle.get(s));
        }*/
    }
    public List<String> getCaseIdColumn() {
        return caseIdColumn;
    }
    public void setCaseIdColumn(List<String> caseIdColumn) {
        this.caseIdColumn = caseIdColumn;
    }
    public List<String> getActivityColumn() {
        return activityColumn;
    }
    public void setActivityColumn(List<String> activityColumn) {
        this.activityColumn = activityColumn;
    }
    public int getSuccessionThreshold() {
        return successionThreshold;
    }
    public void setSuccessionThreshold(int successionThreshold) {
        this.successionThreshold = successionThreshold;
```

```
    }
    public double getDependencyThreshold() {
        return dependencyThreshold;
    }
    public void setDependencyThreshold(double dependencyThreshold) {
        this.dependencyThreshold = dependencyThreshold;
    }
    public int getWindowSize() {
        return windowSize;
    }
    public void setWindowSize(int windowSize) {
        this.windowSize = windowSize;
    }
    public Log getLog() {
        return log;
    }
    public void setLog(Log log) {
        this.log = log;
    }
    public HashMap<String, String> getActivityFullToShortTitle() {
        return activityFullToShortTitle;
    }
    public List<String> getBeginTimestampColumn() {
        return beginTimestampColumn;
    }
    public void setBeginTimestampColumn(List<String> beginTimestampColumn) {
        this.beginTimestampColumn = beginTimestampColumn;
    }
    public List<String> getEndTimestampColumn() {
        return endTimestampColumn;
    }
    public void setEndTimestampColumn(List<String> endTimestampColumn) {
        this.endTimestampColumn = endTimestampColumn;
    }
    public HashMap<Integer, ArrayList<Integer>> getNumericTraces() {
        return numericTraces;
    }
    public void setNumericTraces(HashMap<Integer, ArrayList<Integer>>
numericTraces) {
        this.numericTraces = numericTraces;
    }
    public HashMap<String, String> getActivityShortToFullTitle() {
        return activityShortToFullTitle;
    }
    public boolean isPrepareForSimulation() {
        return prepareForSimulation;
    }
    public void setPrepareForSimulation(boolean prepareForSimulation) {
        this.prepareForSimulation = prepareForSimulation;
    }
    public boolean isLogValid() {
        return isLogValid;
    }
    public void setLogValid(boolean logValid) {
        isLogValid = logValid;
    }
    public String getStartActivity() {
        return startActivity;
    }
    public void setStartActivity(String startActivity) {
        this.startActivity = startActivity;
    }
    public String getEndActivity() {
        return endActivity;
```

```
    }
    public void setEndActivity(String endActivity) {
        this.endActivity = endActivity;
    }
}
package ProcessMining.utils;
public class IntPair{
    private Long id;
    public int x;
    public int y;
    public IntPair(int x, int y){
        this.x = x;
        this.y = y;
        this.id = (long)x*10+y;
    }
    @Override
    public boolean equals(Object v) {
        boolean retVal = false;
        if (v instanceof IntPair){
            retVal = x==((IntPair)v).x && y==((IntPair)v).y;
        }
        return retVal;
    }
    @Override
    public int hashCode() {
        int hash = 2;
        hash = 12 * hash + (this.id != null ? this.id.hashCode() : 0);
        return hash;
    }
}
package ProcessMining.utils;
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class IO {
    private static final char DEFAULT_SEPARATOR = ';';
    private static final char DEFAULT_QUOTE = '"';
    public static List<List<String>> openCSVFile(JFrame frame){
        FileDialog fdlg;
        fdlg = new FileDialog(frame, "Open a file ", FileDialog.LOAD);
        fdlg.setFile("*.csv");
        fdlg.setVisible(true);
        String csvFile = fdlg.getDirectory() + fdlg.getFile();
        Scanner scanner = null;
        List<List<String>> table = new ArrayList<>();
        try {
            scanner = new Scanner(new File(csvFile));
            while (scanner.hasNext()) {
                String s = scanner.nextLine().trim().replaceAll("\\p{C}",
"");
                //System.out.println("!!!!"+s+"!!!");
                List<String> row = parseLine(s);
                if (row.size() >= 4 && !row.get(0).equals("")) {
                    table.add(row);
                } else {
                    break;
                }
                /*for (String s: table.get(table.size()-1)) {
                    System.out.print(s + " ||| ");
                }
```

```
                System.out.println();*/
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        return table;
    }
    public static List<String> parseLine(String cvsLine) {
        return parseLine(cvsLine, DEFAULT_SEPARATOR, DEFAULT_QUOTE);
    }
    public static List<String> parseLine(String cvsLine, char separators) {
        return parseLine(cvsLine, separators, DEFAULT_QUOTE);
    }
    public static List<String> parseLine(String cvsLine, char separators,
char customQuote) {
        List<String> result = new ArrayList<>();
        //if empty, return!
        if (cvsLine == null && cvsLine.isEmpty()) {
            return result;
        }
        if (customQuote == ' ') {
            customQuote = DEFAULT_QUOTE;
        }
        if (separators == ' ') {
            separators = DEFAULT_SEPARATOR;
        }
        StringBuffer curVal = new StringBuffer();
        boolean inQuotes = false;
        boolean startCollectChar = false;
        boolean doubleQuotesInColumn = false;
        char[] chars = cvsLine.toCharArray();
        for (char ch : chars) {
            if (inQuotes) {
                startCollectChar = true;
                if (ch == customQuote) {
                    inQuotes = false;
                    doubleQuotesInColumn = false;
                } else {
                    //Fixed : allow "" in custom quote enclosed
                    if (ch == '\"') {
                        if (!doubleQuotesInColumn) {
                            curVal.append(ch);
                            doubleQuotesInColumn = true;
                        }
                    } else {
                        curVal.append(ch);
                    }
                }
            } else {
                if (ch == customQuote) {
                    inQuotes = true;
                    //Fixed : allow "" in empty quote enclosed
                    if (chars[0] != '"' && customQuote == '\"') {
                        curVal.append('"');
                    }
                    //double quotes in column will hit this!
                    if (startCollectChar) {
                        curVal.append('"');
                    }
                } else if (ch == separators) {
                    result.add(curVal.toString());
                    curVal = new StringBuffer();
                    startCollectChar = false;
```

```
                } else if (ch == '\r') {
                    //ignore LF characters
                    continue;
                } else if (ch == '\n') {
                    //the end, break!
                    break;
                } else {
                    curVal.append(ch);
                }
            }
        }
        result.add(curVal.toString());
        return result;
    }
}
package ProcessMining;
public class Arc {
    private int startPoint;
    private String startPointTitle;
    private int endPoint;
    private String endPointTitle;
    private int successions;
    private double dependency;
    public Arc(int startPoint, String startPointTitle, int endPoint, String
endPointTitle, int successions, double dependency) {
        this.startPoint = startPoint;
        this.startPointTitle = startPointTitle;
        this.endPoint = endPoint;
        this.endPointTitle = endPointTitle;
        this.successions = successions;
        this.dependency = dependency;
    }
    public int getStartPoint() {
        return startPoint;
    }
    public String getStartPointTitle() {
        return startPointTitle;
    }
    public int getEndPoint() {
        return endPoint;
    }
    public String getEndPointTitle() {
        return endPointTitle;
    }
    public int getSuccessions() {
        return successions;
    }
    public double getDependency() {
        return dependency;
    }
    @Override
    public String toString() {
        return startPointTitle + " -> " + endPointTitle + " | " +
successions+"("+Math.round(dependency*100)/100.0+")";
    }
}
package ProcessMining;
import PetriObj.*;
import ProcessMining.utils.InputData;
import ProcessMining.utils.IntPair;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.*;
```

```
import java.util.concurrent.ForkJoinPool;
public class HeuristicMiner {
    private int points_count;
    private int trans_count;
    private ArrayList<PetriP> d_P;
    private ArrayList<PetriT> d_T;
    private ArrayList<ArcIn> d_In;
    private ArrayList<ArcOut> d_Out;
    private int maxDirectSuccessionValue;
    private int maxTraceLength;
    private HashMap<String, Integer> activityFrequency;
    private HashMap<IntPair, Integer> transitionFrequency;
    private HashMap<String, ArrayList<Long>> activityDuration;
    private HashMap<IntPair, ArrayList<Long>> delayDuration;
    private InputData inputData;
    private boolean SHOW_DEBUG_TITLES = false;
    private double DEFAULT_TRANSITION_TIME = 0.0;
    private boolean SHOW_ESS_ELEMENTS = true;
    private double DEFAULT_TIME_BETWEEN_START = 15;
    public PetriNet runConsecutively(InputData inputData) {
        this.inputData = inputData;
        SHOW_ESS_ELEMENTS = inputData.isPrepareForSimulation();
        Log log = inputData.getLog();
        int successionsThreshold = inputData.getSuccessionThreshold();
        double dependencyThreshold = inputData.getDependencyThreshold();
        int windowSize = inputData.getWindowSize();
        maxDirectSuccessionValue = 0;
        maxTraceLength = 0;
        int[][] directSuccessionMatrix =
calculateDirectSuccessionMatrix(log);
        printMatrix(directSuccessionMatrix,
log.getActivityTitleToNumberMap());
        double[][] dependencyMeasuresMatrix =
calculateDependencyMeasuresMatrix(directSuccessionMatrix);
        printMatrix(dependencyMeasuresMatrix,
log.getActivityTitleToNumberMap());
        ArrayList<Arc> approvedArcs = getApprovedArcs(log,
directSuccessionMatrix, successionsThreshold, dependencyMeasuresMatrix,
dependencyThreshold);
        for (Arc arc : approvedArcs){
            System.out.println(arc);
        }
        ArrayList<Node> nodes = getNodes(log, approvedArcs, windowSize);
        for (Node node : nodes){
            System.out.println(node);
        }
        analyse(inputData, log, directSuccessionMatrix,
dependencyMeasuresMatrix, approvedArcs, nodes);
        return generatePetriNet_simplified(approvedArcs, nodes);
    }
    public void analyse(InputData inputData, Log log, int[][]
directSuccessionMatrix, double[][] dependencyMeasuresMatrix, ArrayList<Arc>
approvedArcs, ArrayList<Node> nodes){
        activityFrequency = new HashMap<>();
        transitionFrequency = new HashMap<>();
        for (LogItem item: log.getLogItems()) {
            for (String activity: item.getTrace()) {
                int frequency = item.getFrequency();
                if (activityFrequency.containsKey(activity)) {
                    frequency += activityFrequency.get(activity);
                }
                activityFrequency.put(activity, frequency);
            }
        }
```

```
        for (int i = 0; i < directSuccessionMatrix.length; i++) {
            for (int j = 0; j < directSuccessionMatrix.length; j++) {
                IntPair transition = new IntPair(i,j);
                int frequency = directSuccessionMatrix[i][j];
                if (transitionFrequency.containsKey(transition)) {
                    frequency += transitionFrequency.get(transition);
                }
                transitionFrequency.put(transition, frequency);
            }
        }
        DateTimeFormatter f = DateTimeFormatter.ofPattern("dd.MM.yyyy
HH:mm:ss");
        LocalDateTime begin, end;
        String activity, activity2;
        activityDuration = new HashMap<>();
        delayDuration = new HashMap<>();
        for (int i = 0; i < inputData.getBeginTimestampColumn().size(); i++)
{
            begin =
LocalDateTime.from(f.parse(inputData.getBeginTimestampColumn().get(i)));
            end =
LocalDateTime.from(f.parse(inputData.getEndTimestampColumn().get(i)));
            activity =
inputData.getActivityFullToShortTitle().get(inputData.getActivityColumn().get
(i));
            long dur = Math.abs(ChronoUnit.SECONDS.between(begin, end));
            ArrayList<Long> list = new ArrayList<>();
            if (activityDuration.containsKey(activity)) {
                list = activityDuration.get(activity);
            }
            list.add(dur);
            activityDuration.put(activity, list);
        }
        for (ArrayList<Integer> nums: inputData.getNumericTraces().values())
{
            for (int i = 0; i < nums.size()-1; i++) {
                begin =
LocalDateTime.from(f.parse(inputData.getEndTimestampColumn().get(nums.get(i))
));
                end =
LocalDateTime.from(f.parse(inputData.getBeginTimestampColumn().get(nums.get(i
+1))));
                activity =
inputData.getActivityFullToShortTitle().get(inputData.getActivityColumn().get
(nums.get(i)));
                activity2 =
inputData.getActivityFullToShortTitle().get(inputData.getActivityColumn().get
(nums.get(i+1)));
                IntPair pair = new
IntPair(log.getActivityTitleToNumberMap().get(activity),log.getActivityTitleT
oNumberMap().get(activity2));
                long dur = Math.abs(ChronoUnit.SECONDS.between(begin, end));
                ArrayList<Long> list = new ArrayList<>();
                if (delayDuration.containsKey(pair)) {
                    list = delayDuration.get(pair);
                }
                list.add(dur);
                delayDuration.put(pair, list);
            }
        }
        /*System.out.println("_____________________");
        for (String key: activityFrequency.keySet()) {
            System.out.println(key + " | "+ activityFrequency.get(key));
        }
```

```
        System.out.println("_____________________");
        for (IntPair trans: transitionFrequency.keySet()) {
            System.out.println(trans.x + "-" + trans.y + " | "+
transitionFrequency.get(trans));
        }
        System.out.println("_____________________");
        for (String s: activityDuration.keySet()) {
            System.out.println(s + activityDuration.get(s).toString());
        }
        System.out.println("_____________________");
        for (IntPair s: delayDuration.keySet()) {
            System.out.println(s.x+"-"+s.y +
delayDuration.get(s).toString());
        }
        System.out.println("_____________________");*/
    }
    private void newPoint(String title, int tokens){
        d_P.add(new PetriP(title,tokens));
        points_count++;
    }
    private void newTransition(String title, double time){
        d_T.add(new PetriT(title,time));
        trans_count++;
    }
    private void newInArc(int n, int m){
        d_In.add(new ArcIn(d_P.get(n),d_T.get(m),1));
    }
    private void newOutArc(int n, int m){
        d_Out.add(new ArcOut(d_T.get(n),d_P.get(m),1));
    }
    private PetriNet generatePetriNet_original(ArrayList<Arc> arcs,
ArrayList<Node> nodes){
        d_P = new ArrayList<>();
        d_T = new ArrayList<>();
        d_Out = new ArrayList<>();
        d_In = new ArrayList<>();
        points_count = 0;
        trans_count = 0;
        ArrayList<Integer> prev_p = new ArrayList<>();
        ArrayList<Integer> next_p = new ArrayList<>();
        for (int i = 0; i < nodes.size(); i++) {
            newTransition(nodes.get(i).getTitle(), DEFAULT_TRANSITION_TIME);
            newPoint(i+"_prev", i == 0 ? 1 : 0);
            newInArc(points_count-1, trans_count-1);
            prev_p.add(points_count-1);
            newPoint(i+"_next", 0);
            newOutArc(trans_count-1, points_count-1);
            next_p.add(points_count-1);
        }
        HashMap<HashSet<Integer>,Integer> interm_p = new HashMap<>();
        for (int i = 0; i < nodes.size(); i++) {
            for (HashSet<Integer> set :
nodes.get(i).getInputBindings().keySet()) {
                newTransition("y", DEFAULT_TRANSITION_TIME);
                newOutArc(trans_count - 1, prev_p.get(i));
                int target_t = trans_count - 1;
                for (Integer n: set){
                    int intermPointN = 0;
                    HashSet<Integer> thisArc = new
HashSet<>(Arrays.asList(i,n));
                    if (!interm_p.containsKey(thisArc)) {
                        newPoint("int" + i + "-" + n, 0);
                        interm_p.put(thisArc, points_count - 1);
                        intermPointN = points_count - 1;
```

```
                    } else {
                        intermPointN = interm_p.get(thisArc);
                    }
                    newInArc(intermPointN,target_t);
                }
            }
            for (HashSet<Integer> set :
nodes.get(i).getOutputBindings().keySet()) {
                newTransition("x", DEFAULT_TRANSITION_TIME);
                newInArc(next_p.get(i), trans_count - 1);
                int source_t = trans_count - 1;
                for (Integer n: set){
                    int intermPointN = 0;
                    HashSet<Integer> thisArc = new
HashSet<>(Arrays.asList(i,n));
                    if (!interm_p.containsKey(thisArc)) {
                        newPoint("int" + i + "-" + n, 0);
                        interm_p.put(thisArc, points_count - 1);
                        intermPointN = points_count - 1;
                    } else {
                        intermPointN = interm_p.get(thisArc);
                    }
                    newOutArc(source_t, intermPointN);
                }
            }
        }
        PetriNet net = null;
        try {
            net = new PetriNet("ProcessMinedNet", d_P, d_T, d_In, d_Out);
        } catch (ExceptionInvalidTimeDelay exceptionInvalidTimeDelay) {
            exceptionInvalidTimeDelay.printStackTrace();
        }
        return net;
    }
    private PetriNet generatePetriNet_semisimplified(ArrayList<Arc> arcs,
ArrayList<Node> nodes){
        d_P = new ArrayList<>();
        d_T = new ArrayList<>();
        d_Out = new ArrayList<>();
        d_In = new ArrayList<>();
        points_count = 0;
        trans_count = 0;
        HashMap<HashSet<Integer>,Integer> interm_p = new HashMap<>();
        HashMap<Integer,Integer> prev_p = new HashMap<>();
        HashMap<Integer,Integer> next_p = new HashMap<>();
        HashMap<Integer,Integer> main_t = new HashMap<>();
        for (int i = 0; i < nodes.size(); i++) {
            boolean hasInput = !nodes.get(i).getInputBindings().isEmpty();
            boolean hasOutput = !nodes.get(i).getOutputBindings().isEmpty();
            if (hasInput && hasOutput) {
                newTransition(nodes.get(i).getTitle(),
DEFAULT_TRANSITION_TIME);
                main_t.put(i, trans_count-1);
            } else if (hasInput) {
                // end point
                newTransition(nodes.get(i).getTitle(),
DEFAULT_TRANSITION_TIME);
                main_t.put(i, trans_count-1);
                newPoint("END", 0);
                newOutArc(trans_count-1, points_count-1);
            } else {
                // start point
                newTransition(nodes.get(i).getTitle(),
DEFAULT_TRANSITION_TIME);
```

```
                main_t.put(i, trans_count-1);
                newPoint("START", 1);
                newInArc(points_count-1, trans_count-1);
            }
        }
        for (int i = 0; i < nodes.size(); i++) {
            boolean simpleInput = nodes.get(i).getInputBindings().size() <=
1;
            boolean simpleOutput = nodes.get(i).getOutputBindings().size() <=
1;
            if (!simpleInput){
                //newTransition("", DEFAULT_TRANSITION_TIME);
                newPoint("", 0);
                newInArc(points_count-1, main_t.get(i));
                prev_p.put(i, points_count-1);
            }
            if (!simpleOutput){
                //newTransition("", DEFAULT_TRANSITION_TIME);
                newPoint("", 0);
                newOutArc(main_t.get(i), points_count-1);
                next_p.put(i, points_count-1);
            }
            for (HashSet<Integer> set :
nodes.get(i).getInputBindings().keySet()) {
                if (simpleInput) {
                    for (Integer n : set) {
                        int intermPointN = 0;
                        HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                        if (!interm_p.containsKey(theseTwoSet)) {
                            newPoint(nodes.get(i).getTitle() + "-" +
nodes.get(n).getTitle(), 0);
                            intermPointN = points_count - 1;
                            interm_p.put(theseTwoSet, intermPointN);
                        } else {
                            intermPointN = interm_p.get(theseTwoSet);
                        }
                        newInArc(intermPointN, main_t.get(i));
                    }
                } else {
                    newTransition("", DEFAULT_TRANSITION_TIME);
                    int target_t = trans_count - 1;
                    newOutArc(target_t, prev_p.get(i));
                    for (Integer n : set) {
                        int intermPointN = 0;
                        HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                        if (!interm_p.containsKey(theseTwoSet)) {
                            newPoint(nodes.get(i).getTitle() + "-" +
nodes.get(n).getTitle(), 0);
                            intermPointN = points_count - 1;
                            interm_p.put(theseTwoSet, intermPointN);
                        } else {
                            intermPointN = interm_p.get(theseTwoSet);
                        }
                        newInArc(intermPointN, target_t);
                    }
                }
            }
            for (HashSet<Integer> set :
nodes.get(i).getOutputBindings().keySet()) {
                if (simpleOutput) {
                    for (Integer n : set) {
                        int intermPointN = 0;
```

```
                        HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                        if (!interm_p.containsKey(theseTwoSet)) {
                            newPoint(nodes.get(i).getTitle() + "-" +
nodes.get(n).getTitle(), 0);
                            intermPointN = points_count - 1;
                            interm_p.put(theseTwoSet, intermPointN);
                        } else {
                            intermPointN = interm_p.get(theseTwoSet);
                        }
                        newOutArc(main_t.get(i), intermPointN);
                    }
                } else {
                    newTransition("", DEFAULT_TRANSITION_TIME);
                    int source_t = trans_count - 1;
                    newInArc(next_p.get(i), source_t);
                    for (Integer n : set) {
                        int intermPointN = 0;
                        HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                        if (!interm_p.containsKey(theseTwoSet)) {
                            newPoint(nodes.get(i).getTitle() + "-" +
nodes.get(n).getTitle(), 0);
                            intermPointN = points_count - 1;
                            interm_p.put(theseTwoSet, intermPointN);
                        } else {
                            intermPointN = interm_p.get(theseTwoSet);
                        }
                        newOutArc(source_t, intermPointN);
                    }
                }
            }
        }
        PetriNet net = null;
        try {
            net = new PetriNet("ProcessMinedNet", d_P, d_T, d_In, d_Out);
        } catch (ExceptionInvalidTimeDelay exceptionInvalidTimeDelay) {
            exceptionInvalidTimeDelay.printStackTrace();
        }
        return net;
    }
    private PetriNet generatePetriNet_simplified(ArrayList<Arc> arcs,
ArrayList<Node> nodes){
        d_P = new ArrayList<>();
        d_T = new ArrayList<>();
        d_Out = new ArrayList<>();
        d_In = new ArrayList<>();
        points_count = 0;
        trans_count = 0;
        PetriNet net = null;
        HashMap<HashSet<Integer>,Integer> interm_p = new HashMap<>();
        HashMap<Integer,Integer> prev_p = new HashMap<>();
        HashMap<Integer,Integer> next_p = new HashMap<>();
        HashMap<Integer,Integer> main_t = new HashMap<>();
        int start_count = 0;
        int end_count = 0;
        HashMap<HashSet<Integer>,Boolean> simplifiedArcs = new HashMap<>();
        HashSet<HashSet<Integer>> mustBeSimplifiedArcs = new HashSet<>();
        HashMap<IntPair, Double> avgDelayDuration = new HashMap<>();
        HashMap<Integer, Double> avgActivityDuration = new HashMap<>();
        for (IntPair p: delayDuration.keySet()) {
            List<Long> list = delayDuration.get(p);
            double avg = 0;
            Long sum = 0L;
```

```
            if (list.size() > 0) {
                for (Long l : list) {
                    sum += l;
                }
                avg = Math.round(sum * 1f / list.size());
                avgDelayDuration.put(p, avg);
            }
        }
        int startTransN =
inputData.getLog().getActivityTitleToNumberMap().get(inputData.getStartActivi
ty());
        int finishTransN =
inputData.getLog().getActivityTitleToNumberMap().get(inputData.getEndActivity
());
        try {
            for (int i = 0; i < nodes.size(); i++) {
                boolean hasInput =
!nodes.get(i).getInputBindings().isEmpty();
                boolean hasOutput =
!nodes.get(i).getOutputBindings().isEmpty();
                List<Long> list =
activityDuration.get(nodes.get(i).getTitle());
                double avg = 0;
                Long sum = 0L;
                if (list.size() > 0) {
                    for (Long l : list) {
                        sum += l;
                    }
                    avg = Math.round(sum * 1f / list.size());
                    avgActivityDuration.put(i, avg);
                }
                if (i == startTransN && hasOutput) {
                    // start point
                    if (SHOW_ESS_ELEMENTS) {
                        newTransition(nodes.get(i).getTitle(), avg);
                        main_t.put(i, trans_count - 1);
                        newPoint("START", 0);
                        newInArc(points_count - 1, trans_count - 1);
                        start_count++;
                        newTransition("", DEFAULT_TIME_BETWEEN_START);
                        newOutArc(trans_count - 1, points_count - 1);
                        newPoint("", 1);
                        newOutArc(trans_count - 1, points_count - 1);
                        newInArc(points_count - 1, trans_count - 1);
                    } else {
                        newTransition(nodes.get(i).getTitle(), avg);
                        main_t.put(i, trans_count - 1);
                        newPoint("START", 1);
                        newInArc(points_count - 1, trans_count - 1);
                        start_count++;
                    }
                } else if (i == finishTransN && hasInput) {
                    // end point
                    newTransition(nodes.get(i).getTitle(), avg);
                    main_t.put(i, trans_count - 1);
                    newPoint("END", 0);
                    newOutArc(trans_count - 1, points_count - 1);
                    end_count++;
                } else if (hasInput && hasOutput) {
                    newTransition(nodes.get(i).getTitle(), avg);
                    main_t.put(i, trans_count - 1);
                } else {
                    System.out.println("Activity " + nodes.get(i).getTitle()
+ " is NOT included");
```

```
                }
                boolean simpleInput = nodes.get(i).getInputBindings().size()
<= 1;
                boolean simpleOutput =
nodes.get(i).getOutputBindings().size() <= 1;
                if (!simpleInput) {
                    newPoint(SHOW_DEBUG_TITLES ? "onNotSimpleIn" +
nodes.get(i).getTitle() : "", 0);
                    newInArc(points_count - 1, main_t.get(i));
                    prev_p.put(i, points_count - 1);
                }
                if (!simpleOutput) {
                    newPoint(SHOW_DEBUG_TITLES ? "onNotSimpleOut" +
nodes.get(i).getTitle() : "", 0);
                    newOutArc(main_t.get(i), points_count - 1);
                    next_p.put(i, points_count - 1);
                }
                HashMap<Integer, Integer> outputNodesQuantities = new
HashMap<>();
                HashMap<Integer, Integer> inputNodesQuantities = new
HashMap<>();
                for (HashSet<Integer> set :
nodes.get(i).getInputBindings().keySet()) {
                    for (Integer n : set) {
                        int quantity = set.size() > 1 ? 2 : 1;
                        if (inputNodesQuantities.containsKey(n)) {
                            quantity += inputNodesQuantities.get(n);
                        }
                        inputNodesQuantities.put(n, quantity);
                    }
                }
                for (Integer j : inputNodesQuantities.keySet()) {
                    if (inputNodesQuantities.get(j) == 1 && !simpleInput) {
                        //System.out.println("SIMPLIFIED IN:" +
nodes.get(i).getTitle() + "-" + nodes.get(j).getTitle());
                        mustBeSimplifiedArcs.add(new
HashSet<>(Arrays.asList(i, j)));
                    }
                }
                for (HashSet<Integer> set :
nodes.get(i).getOutputBindings().keySet()) {
                    for (Integer n : set) {
                        int quantity = set.size() > 1 ? 2 : 1;
                        if (outputNodesQuantities.containsKey(n)) {
                            quantity += outputNodesQuantities.get(n);
                        }
                        outputNodesQuantities.put(n, quantity);
                    }
                }
                for (Integer j : outputNodesQuantities.keySet()) {
                    if (outputNodesQuantities.get(j) == 1 && !simpleOutput) {
                        //System.out.println("SIMPLIFIED OUT:" +
nodes.get(i).getTitle() + "-" + nodes.get(j).getTitle());
                        mustBeSimplifiedArcs.add(new
HashSet<>(Arrays.asList(i, j)));
                    }
                }
            }
            for (int i = 0; i < nodes.size(); i++) {
                boolean simpleInput = nodes.get(i).getInputBindings().size()
<= 1;
                boolean simpleOutput =
nodes.get(i).getOutputBindings().size() <= 1;
                for (HashSet<Integer> set :
```

```
nodes.get(i).getInputBindings().keySet()) {
                    if (simpleInput) {
                        int firstN = set.iterator().hasNext() ?
set.iterator().next() : -1;
                        if (firstN != -1 && !(set.size() == 1 &&
mustBeSimplifiedArcs.contains(new HashSet<>(Arrays.asList(i, firstN))))) {
                            for (Integer n : set) {
                                int intermPointN = 0;
                                HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                                if (!interm_p.containsKey(theseTwoSet)) {
                                    newPoint(SHOW_DEBUG_TITLES ?
"intermSimpleInput" + nodes.get(i).getTitle() : "", 0);
                                    intermPointN = points_count - 1;
                                    interm_p.put(theseTwoSet, intermPointN);
                                } else {
                                    intermPointN = interm_p.get(theseTwoSet);
                                }
                                newInArc(intermPointN, main_t.get(i));
                            }
                        } else if (firstN != -1 &&
!simplifiedArcs.containsKey(new HashSet<>(Arrays.asList(i, firstN)))) {
                            if (prev_p.get(i) != null) {
                                newOutArc(main_t.get(firstN), prev_p.get(i));
                                simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                            } else if (next_p.get(firstN) != null) {
                                newInArc(next_p.get(firstN), main_t.get(i));
                                simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                            }
                        }
                    } else {
                        int firstN = set.iterator().hasNext() ?
set.iterator().next() : -1;
                        if (firstN != -1 && !(set.size() == 1 &&
mustBeSimplifiedArcs.contains(new HashSet<>(Arrays.asList(i, firstN))))) {
                            double time = avgDelayDuration.containsKey(new
IntPair(firstN, i)) ? avgDelayDuration.get(new IntPair(firstN, i)) :
DEFAULT_TRANSITION_TIME;
                            newTransition(SHOW_DEBUG_TITLES ?
(nodes.get(firstN).getTitle() + "-" + nodes.get(i).getTitle()) : "", time);
                            int target_t = trans_count - 1;
                            newOutArc(target_t, prev_p.get(i));
                            for (Integer n : set) {
                                int intermPointN = 0;
                                HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                                if (next_p.containsKey(n)) {
                                    intermPointN = next_p.get(n);
                                } else if
(!interm_p.containsKey(theseTwoSet)) {
                                    newPoint(SHOW_DEBUG_TITLES ?
"intermInput" + nodes.get(i).getTitle() : "", 0);
                                    intermPointN = points_count - 1;
                                    interm_p.put(theseTwoSet, intermPointN);
                                } else {
                                    intermPointN = interm_p.get(theseTwoSet);
                                }
                                newInArc(intermPointN, target_t);
                            }
                        } else if (firstN != -1 &&
!simplifiedArcs.containsKey(new HashSet<>(Arrays.asList(i, firstN)))) {
                            if (nodes.get(firstN).getOutputBindings().size()
```

```
<= 1) {
                                    newOutArc(main_t.get(firstN), prev_p.get(i));
                                    //double time =
avgDelayDuration.containsKey(new IntPair(firstN, i)) ?
avgActivityDuration.get(firstN)+avgDelayDuration.get(new IntPair(firstN, i))
: DEFAULT_TRANSITION_TIME;
                                    //d_T.get(main_t.get(i)).setParametr(time);
                                    simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                                } else {
                                    if (next_p.get(firstN) == null) {
                                        newPoint(SHOW_DEBUG_TITLES ?
"simplifiedInput" + nodes.get(i).getTitle() : "", 0);
                                        next_p.put(firstN, points_count - 1);
                                    }
                                    double time =
avgDelayDuration.containsKey(new IntPair(firstN, i)) ?
avgDelayDuration.get(new IntPair(firstN, i)) : DEFAULT_TRANSITION_TIME;
                                    newTransition(SHOW_DEBUG_TITLES ?
(nodes.get(firstN).getTitle() + "-" + nodes.get(i).getTitle()) : "", time);
                                    newOutArc(trans_count - 1, prev_p.get(i));
                                    newInArc(next_p.get(firstN), trans_count -
1);
                                    simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                                }
                            }
                        }
                    }
                    for (HashSet<Integer> set :
nodes.get(i).getOutputBindings().keySet()) {
                        if (simpleOutput) {
                            int firstN = set.iterator().hasNext() ?
set.iterator().next() : -1;
                            if (firstN != -1 && !(set.size() == 1 &&
mustBeSimplifiedArcs.contains(new HashSet<>(Arrays.asList(i, firstN))))) {
                                for (Integer n : set) {
                                    int intermPointN = 0;
                                    HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                                    if (!interm_p.containsKey(theseTwoSet)) {
                                        newPoint(SHOW_DEBUG_TITLES ?
"intermSimpleOutput" + nodes.get(i).getTitle() : "", 0);
                                        //System.out.println("_____________
"+"intermSimpleOutput"+nodes.get(i).getTitle());
                                        intermPointN = points_count - 1;
                                        interm_p.put(theseTwoSet, intermPointN);
                                    } else {
                                        intermPointN = interm_p.get(theseTwoSet);
                                    }
                                    newOutArc(main_t.get(i), intermPointN);
                                }
                            } else if (firstN != -1 &&
!simplifiedArcs.containsKey(new HashSet<>(Arrays.asList(i, firstN)))) {
                                if (next_p.get(i) != null) {
                                    newInArc(next_p.get(i), main_t.get(firstN));
                                    simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                                } else if (prev_p.get(firstN) != null) {
                                    newOutArc(main_t.get(i), prev_p.get(firstN));
                                    simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                                }
                            }
```

```
                    } else {
                        int firstN = set.iterator().hasNext() ?
set.iterator().next() : -1;
                        if (firstN != -1 && !(set.size() == 1 &&
mustBeSimplifiedArcs.contains(new HashSet<>(Arrays.asList(i, firstN))))) {
                            double time = avgDelayDuration.containsKey(new
IntPair(i, firstN)) ? avgDelayDuration.get(new IntPair(i, firstN)) :
DEFAULT_TRANSITION_TIME;
                            newTransition(SHOW_DEBUG_TITLES ?
(nodes.get(i).getTitle() + "-" + nodes.get(firstN).getTitle()) : "", time);
                            int source_t = trans_count - 1;
                            newInArc(next_p.get(i), source_t);
                            for (Integer n : set) {
                                int intermPointN = 0;
                                HashSet<Integer> theseTwoSet = new
HashSet<>(Arrays.asList(i, n));
                                if (prev_p.containsKey(n)) {
                                    intermPointN = prev_p.get(n);
                                } else if
(!interm_p.containsKey(theseTwoSet)) {
                                    newPoint(SHOW_DEBUG_TITLES ?
"intermOutput" + nodes.get(i).getTitle() : "", 0);
                                    intermPointN = points_count - 1;
                                    interm_p.put(theseTwoSet, intermPointN);
                                } else {
                                    intermPointN = interm_p.get(theseTwoSet);
                                }
                                newOutArc(source_t, intermPointN);
                            }
                        } else if (firstN != -1 &&
!simplifiedArcs.containsKey(new HashSet<>(Arrays.asList(i, firstN)))) {
                            if (nodes.get(firstN).getInputBindings().size()
<= 1) {
                                newInArc(next_p.get(i), main_t.get(firstN));
                                //double time =
avgDelayDuration.containsKey(new IntPair(i, firstN)) ?
avgActivityDuration.get(firstN)+avgDelayDuration.get(new IntPair(i, firstN))
: DEFAULT_TRANSITION_TIME;

//d_T.get(main_t.get(firstN)).setParametr(time);
                                simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                            } else {
                                if (prev_p.get(firstN) == null) {
                                    newPoint(SHOW_DEBUG_TITLES ?
"simplifiedOutput" + nodes.get(i).getTitle() : "", 0);
                                    prev_p.put(firstN, points_count - 1);
                                }
                                double time =
avgDelayDuration.containsKey(new IntPair(i, firstN)) ?
avgDelayDuration.get(new IntPair(i, firstN)) : DEFAULT_TRANSITION_TIME;
                                newTransition(SHOW_DEBUG_TITLES ?
(nodes.get(i).getTitle() + "-" + nodes.get(firstN).getTitle()) : "", time);
                                newInArc(next_p.get(i), trans_count - 1);
                                newOutArc(trans_count - 1,
prev_p.get(firstN));
                                simplifiedArcs.put(new
HashSet<>(Arrays.asList(i, firstN)), true);
                            }
                        }
                    }
                }
            }
        } catch (Exception e) {
```

```
            return net;
        }
        if (SHOW_ESS_ELEMENTS) {
            for (int tr: main_t.values()) {
                newPoint("", 1);
                newOutArc(tr, points_count-1);
                newInArc(points_count-1,tr);
            }
        }
        try {
            if (d_T.isEmpty()) {
                if (SHOW_ESS_ELEMENTS) {
                    newTransition("", DEFAULT_TRANSITION_TIME);
                    newPoint("START", 0);
                    newInArc(points_count - 1, trans_count - 1);
                    newTransition("", DEFAULT_TIME_BETWEEN_START);
                    newOutArc(trans_count - 1, points_count - 1);
                    newPoint("", 1);
                    newOutArc(trans_count - 1, points_count - 1);
                    newInArc(points_count - 1, trans_count - 1);
                } else {
                    newTransition("", DEFAULT_TRANSITION_TIME);
                    newPoint("START", 1);
                    newInArc(points_count - 1, trans_count - 1);
                }
                newPoint("END", 0);
                newOutArc(0, points_count - 1);
                start_count++;
                end_count++;
            }
            if (start_count == 1 && end_count == 1){
                net = new PetriNet("ProcessMinedNet", d_P, d_T, d_In, d_Out);
            }
        } catch (ExceptionInvalidTimeDelay exceptionInvalidTimeDelay) {
            exceptionInvalidTimeDelay.printStackTrace();
        }
        return net;
    }
    private ArrayList<Node> getNodes(Log log, ArrayList<Arc> approvedArcs,
int windowSize) {
        Node[] nodesArray = new
Node[log.getActivityTitleToNumberMap().keySet().size()];
        for (String title : log.getActivityTitleToNumberMap().keySet()) {
            int activityN = log.getActivityTitleToNumberMap().get(title);
            HashSet<Integer> activitiesBefore = new HashSet<>();
            HashSet<Integer> activitiesAfter = new HashSet<>();
            for (Arc arc : approvedArcs) {
                if (activityN == arc.getEndPoint())
                    activitiesBefore.add(arc.getStartPoint());
                if (activityN == arc.getStartPoint())
                    activitiesAfter.add(arc.getEndPoint());
            }
            nodesArray[activityN] = new Node(activityN, title,
activitiesBefore, activitiesAfter);
        }
        for (LogItem logItem : log.getLogItems()) {
            final int MAX_SIZE = logItem.getTrace().size();
            for (int i = 0; i < MAX_SIZE; i++) {
                int currentActivityN =
log.getActivityTitleToNumberMap().get(logItem.getTrace().get(i));
                HashSet<Integer> before = new HashSet<>();
                HashSet<Integer> after = new HashSet<>();
                int beforeStart = i - 1 < 0 ? 0 : i - 1;
                int beforeEnd = i - windowSize < 0 ? 0 : i - windowSize;
```

```
                int afterStart = i + 1 > MAX_SIZE - 1 ? MAX_SIZE - 1 : i + 1;
                int afterEnd = i + windowSize > MAX_SIZE - 1 ? MAX_SIZE - 1 :
i + windowSize;
                //before
                HashSet<Integer> beforeWindow = new HashSet<>();
                for (int j = beforeStart; j >= beforeEnd; j--) {
                    int beforeActivityN =
log.getActivityTitleToNumberMap().get(logItem.getTrace().get(j));
                    beforeWindow.add(beforeActivityN);
                }
                for (int activity :
nodesArray[currentActivityN].getActivitiesBefore()) {
                    if (beforeWindow.contains(activity))
                        before.add(activity);
                }
                int n = 0;
                if (before.size() > 0) {
                    HashSet<Integer> remove = new HashSet<>();
                    for (int activity1 : before) {
                        for (int activity2 : before) {
                            for (Arc arc : approvedArcs) {
                                if (arc.getStartPoint() == activity1 &&
arc.getEndPoint() == activity2 && activity1 != activity2) {
                                    remove.add(activity1);
                                }
                            }
                        }
                    }
                    for (int a : remove) {
                        before.remove(a);
                    }
                    if
(nodesArray[currentActivityN].getInputBindings().keySet().contains(before)) {
                        n =
nodesArray[currentActivityN].getInputBindings().get(before);
                    }

nodesArray[currentActivityN].getInputBindings().put(before, n +
logItem.getFrequency());
                }
                // after
                HashSet<Integer> afterWindow = new HashSet<>();
                for (int j = afterStart; j <= afterEnd; j++) {
                    int afterActivityN =
log.getActivityTitleToNumberMap().get(logItem.getTrace().get(j));
                    afterWindow.add(afterActivityN);
                }
                for (int activity :
nodesArray[currentActivityN].getActivitiesAfter()) {
                    if (afterWindow.contains(activity))
                        after.add(activity);
                }
                n = 0;
                if (after.size() > 0) {
                    HashSet<Integer> remove = new HashSet<>();
                    for (int activity1 : after) {
                        for (int activity2 : after) {
                            for (Arc arc : approvedArcs) {
                                if (arc.getStartPoint() == activity1 &&
arc.getEndPoint() == activity2 && activity1 != activity2) {
                                    remove.add(activity2);
                                }
                            }
                        }
```

```
                    }
                    for (int a : remove) {
                        after.remove(a);
                    }
                    if
(nodesArray[currentActivityN].getOutputBindings().keySet().contains(after)) {
                        n =
nodesArray[currentActivityN].getOutputBindings().get(after);
                    }

nodesArray[currentActivityN].getOutputBindings().put(after, n +
logItem.getFrequency());
                }
                // node frequencies

nodesArray[currentActivityN].incrementFrequency(logItem.getFrequency());
            }
        }
        ArrayList<Node> nodesList = new ArrayList<>();
        nodesList.addAll(Arrays.asList(nodesArray));
        return nodesList;
    }
    private ArrayList<Arc> getApprovedArcs(Log log, int[][]
directSuccessionMatrix, int successionsThreshold, double[][]
dependencyMeasuresMatrix, double dependencyThreshold) {
        ArrayList<Arc> arcs = new ArrayList<>();
        for (String startTitle : log.getActivityTitleToNumberMap().keySet())
{
            int startN = log.getActivityTitleToNumberMap().get(startTitle);
            for (String endTitle :
log.getActivityTitleToNumberMap().keySet()) {
                int endN = log.getActivityTitleToNumberMap().get(endTitle);
                if (directSuccessionMatrix[startN][endN] >=
successionsThreshold &&
                        dependencyMeasuresMatrix[startN][endN] >=
dependencyThreshold) {
                    arcs.add(new
Arc(startN,startTitle,endN,endTitle,directSuccessionMatrix[startN][endN],depe
ndencyMeasuresMatrix[startN][endN]));
                }
            }
        }
        return arcs;
    }
    public int[][] calculateDirectSuccessionMatrix(Log log) {
        maxDirectSuccessionValue = 0;
        maxTraceLength = 0;
        int activityCount = log.getActivityCount();
        int[][] matrix = new int[activityCount][activityCount];
        for (LogItem logItem: log.getLogItems()) {
            if (logItem.getTrace().size() > 1) {
                for (int i = 0; i < logItem.getTrace().size() - 1; i++) {
                    int row =
log.getActivityTitleToNumberMap().get(logItem.getTrace().get(i));
                    int col =
log.getActivityTitleToNumberMap().get(logItem.getTrace().get(i + 1));
                    matrix[row][col] += logItem.getFrequency();
                    if (matrix[row][col] > maxDirectSuccessionValue) {
                        maxDirectSuccessionValue = matrix[row][col];
                    }
                }
            }
            if (logItem.getTrace().size() > maxTraceLength){
                maxTraceLength = logItem.getTrace().size();
```

```
            }
        }
        return matrix;
    }
    private double[][] calculateDependencyMeasuresMatrix(int[][]
successionMatrix) {
        int activityCount = successionMatrix.length;
        double[][] matrix = new double[activityCount][activityCount];
        for (int i = 0; i < activityCount; i++) {
            for (int j = i; j < activityCount; j++) {
                int a = successionMatrix[i][j];
                int b = successionMatrix[j][i];
                if (i == j) {
                    matrix[i][j] = a / (double)(a + 1);
                } else {
                    matrix[i][j] = (a-b) / (double)(a + b + 1);
                    matrix[j][i] = -matrix[i][j];
                }
            }
        }
        return matrix;
    }
    private void printMatrix(double[][] matrix, HashMap<String, Integer>
map){
        System.out.print(String.format( "%2s",""));
        for (String title : map.keySet())
            System.out.print(String.format( "%7s", title));
        System.out.println();
        for (String row : map.keySet()) {
            System.out.print(String.format("%2s", row));
            for (String col : map.keySet()) {
                int i = map.get(row);
                int j = map.get(col);
                System.out.print(String.format("%7s",
Math.round(matrix[i][j]*100)/100.0));
            }
            System.out.println();
        }
    }
    private void printMatrix(int[][] matrix, HashMap<String, Integer> map){
        System.out.print(String.format( "%2s",""));
        for (String title : map.keySet())
            System.out.print(String.format( "%7s", title));
        System.out.println();
        for (String row : map.keySet()) {
            System.out.print(String.format("%2s", row));
            for (String col : map.keySet()) {
                int i = map.get(row);
                int j = map.get(col);
                System.out.print(String.format("%7s", matrix[i][j]));
            }
            System.out.println();
        }
    }
    public int getMaxDirectSuccessionValue() {
        return maxDirectSuccessionValue;
    }
    public int getMaxTraceLength() {
        return maxTraceLength;
    }
    public HashMap<String, Integer> getActivityFrequency() {
        return activityFrequency;
    }
    public HashMap<IntPair, Integer> getTransitionFrequency() {
```

```
        return transitionFrequency;
    }
    public HashMap<String, ArrayList<Long>> getActivityDuration() {
        return activityDuration;
    }
    public HashMap<IntPair, ArrayList<Long>> getDelayDuration() {
        return delayDuration;
    }
    public InputData getInputData() {
        return inputData;
    }
    public void setInputData(InputData inputData) {
        this.inputData = inputData;
    }
}
package ProcessMining;
import java.util.ArrayList;
import java.util.HashMap;
public class Log{
    private ArrayList<LogItem> logItems;
    private HashMap<String, Integer> activityTitleToNumberMap;
    private HashMap<Integer, String> activityNumberToTitleMap;
    public Log(){
        this.logItems = new ArrayList<>();
        this.activityTitleToNumberMap = new HashMap<>();
        this.activityNumberToTitleMap = new HashMap<>();
    }
    public void add(LogItem logItem) {
        logItems.add(logItem);
        for (String title: logItem.getTrace()){
            if (!activityTitleToNumberMap.keySet().contains(title)) {
                int size = activityTitleToNumberMap.size();
                activityTitleToNumberMap.put(title, size);
                activityNumberToTitleMap.put(size, title);
            }
        }
    }
    public ArrayList<LogItem> getLogItems() {
        return logItems;
    }
    public HashMap<String, Integer> getActivityTitleToNumberMap() {
        return activityTitleToNumberMap;
    }
    public int getActivityCount(){
        return getActivityTitleToNumberMap().size();
    }
    public HashMap<Integer, String> getActivityNumberToTitleMap() {
        return activityNumberToTitleMap;
    }
}
package ProcessMining;
import java.util.ArrayList;
import java.util.Arrays;
public class LogItem {
    private ArrayList<String> trace;
    private int frequency;
    public LogItem(String traceString, int frequency){
        this.frequency = frequency;
        this.trace = new ArrayList<>(Arrays.asList(traceString.split(" ")));
    }
    public ArrayList<String> getTrace() {
        return trace;
    }
    public String getTraceNthItem(int n) {
```

```
        if (n >=0 && n < trace.size())
            return trace.get(n);
        else
            return "?";
    }
    public int getFrequency() {
        return frequency;
    }
    @Override
    public String toString() {
        String result = "{";
        for (int i = 0; i < trace.size(); i++) {
            result += trace.get(i);
        }
        result += "} <" + frequency + ">";
        return result;
    }
}
package ProcessMining;
import java.util.HashMap;
import java.util.HashSet;
public class Node {
    private int number;
    private String title;
    private int frequency;
    private HashMap<HashSet<Integer>,Integer> inputBindings;
    private HashMap<HashSet<Integer>,Integer> outputBindings;
    private HashSet<Integer> activitiesBefore;
    private HashSet<Integer> activitiesAfter;
    public Node(int number, String title, HashSet<Integer> activitiesBefore,
HashSet<Integer> activitiesAfter) {
        this.number = number;
        this.title = title;
        this.activitiesBefore = activitiesBefore;
        this.activitiesAfter = activitiesAfter;
        this.inputBindings = new HashMap<>();
        this.outputBindings = new HashMap<>();
    }
    public HashMap<HashSet<Integer>, Integer> getInputBindings() {
        return inputBindings;
    }
    public HashMap<HashSet<Integer>, Integer> getOutputBindings() {
        return outputBindings;
    }
    public HashSet<Integer> getActivitiesBefore() {
        return activitiesBefore;
    }
    public HashSet<Integer> getActivitiesAfter() {
        return activitiesAfter;
    }
    @Override
    public String toString() {
        String result = title + "("+ number + ")=" + frequency + " | Input: "
+ inputBindings;
        for (HashSet<Integer> set : inputBindings.keySet()) {
            //result += "| "+ set + " (" + inputBindings.get(set) + ")|";
        }
        result += " | Output: " + outputBindings;
        for (HashSet<Integer> set : outputBindings.keySet()) {
            //result += "| "+ set + " (" + outputBindings.get(set) + ")|";
```

```
        }
        return result;
    }
    public int getNumber() {
        return number;
    }
    public String getTitle() {
        return title;
    }
    public boolean isEndNode(){
        return outputBindings.size()==0;
    }
    public boolean isStartNode(){
        return inputBindings.size()==0;
    }
    public int getFrequency() {
        return frequency;
    }
    public void incrementFrequency(int n){
        frequency += n;
    }
    public void setFrequency(int frequency) {
        this.frequency = frequency;
    }
    public void addInputBindings(HashMap<HashSet<Integer>,Integer> add){
        for (HashSet<Integer> hs2 : add.keySet()) {
            int n = add.get(hs2);
            if (inputBindings.keySet().contains(hs2)) {
                n += inputBindings.get(hs2);
            }
            inputBindings.put(hs2,n);
        }
    }
    public void addOutputBindings(HashMap<HashSet<Integer>,Integer> add){
        for (HashSet<Integer> hs2 : add.keySet()) {
            int n = add.get(hs2);
            if (outputBindings.keySet().contains(hs2)) {
                n += outputBindings.get(hs2);
            }
            outputBindings.put(hs2,n);
        }
    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
_____ *І.В. Стеценко*	_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“16” квітня 2019 р.	“17” квітня 2019 р.

Аналіз та моделювання дискретно-подійного процесу на основі журналу подій

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5123.1181-с.ТЗ*

на 15 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Розроблена система отримала наступне найменування: «Програмний продукт з аналізу та моделювання дискретно-подійного процесу на основі журналу подій».

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовник: КПІ ім. Ігоря Сікорського, кафедра АСОІУ.

Виконавець: Рябцев Сергій Вячеславович, студент групи ІС-51, ФІОТ.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створенні проектно-експлуатаційної документації виконавець повинен керуватись вимогами наступних нормативних документів:

- ДСТУ 34.201-89 Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем;
- ДСТУ 34.601-90 Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 19.201-78 Технічне завдання. Вимоги до змісту і оформлення.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Планова дата початку роботи зі створення системи – 01.02.2019.

Планова дата закінчення роботи зі створення системи – 03.06.2019.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Запропонований програмний продукт призначений для виявлення, аналізу та візуалізації моделі дискретно-подійного процесу на основі журналу подій.

2.2 Цілі створення системи

Мета роботи – розробка, дослідження та використання реалізації евристичного алгоритму виявлення процесу (англ. Heuristic Miner) для аналізу та моделювання дискретно-подійного процесу.

Для реалізації поставленої мети роботи необхідно розв’язати наступні задачі:

- розробка та реалізація евристичного алгоритму виявлення моделі процесів;
- розробка та реалізація алгоритму перетворення виявленого процесу у мережу Петрі;
- створення засобів проведення аналізу моделі дискретно-подійного процесу на основі журналу подій;
- створення засобів графічної візуалізації виявленої моделі та результатів її аналізу.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Опис процесу діяльності представлено на рисунку 3.1 у вигляді діаграми діяльності. Уся діяльність розподілена між двома ролями: користувача та програмного продукту.

Рисунок 3.1 – Схема структурна діяльності

Як можна зазначити з діаграми діяльності, користувач завантажує вхідні дані, задає параметри виявлення моделі процесу. Після цього програмний продукт виявляє дискретно-подійний процес та візуалізує його у вигляді мережі Петрі. Користувач переглядає дану візуалізацію. Потім програмний продукт аналізує виявлений дискретно-подійний процес та знову ж таки візуалізує його для користувача. Користувач, в свою чергу, переглядає звіт з аналізу процесу та завершує роботу з програмним продуктом.

Специфікацію функціональної поведінки системи представлена у вигляді діаграми варіантів використання, що зображена на рисунку 3.2.

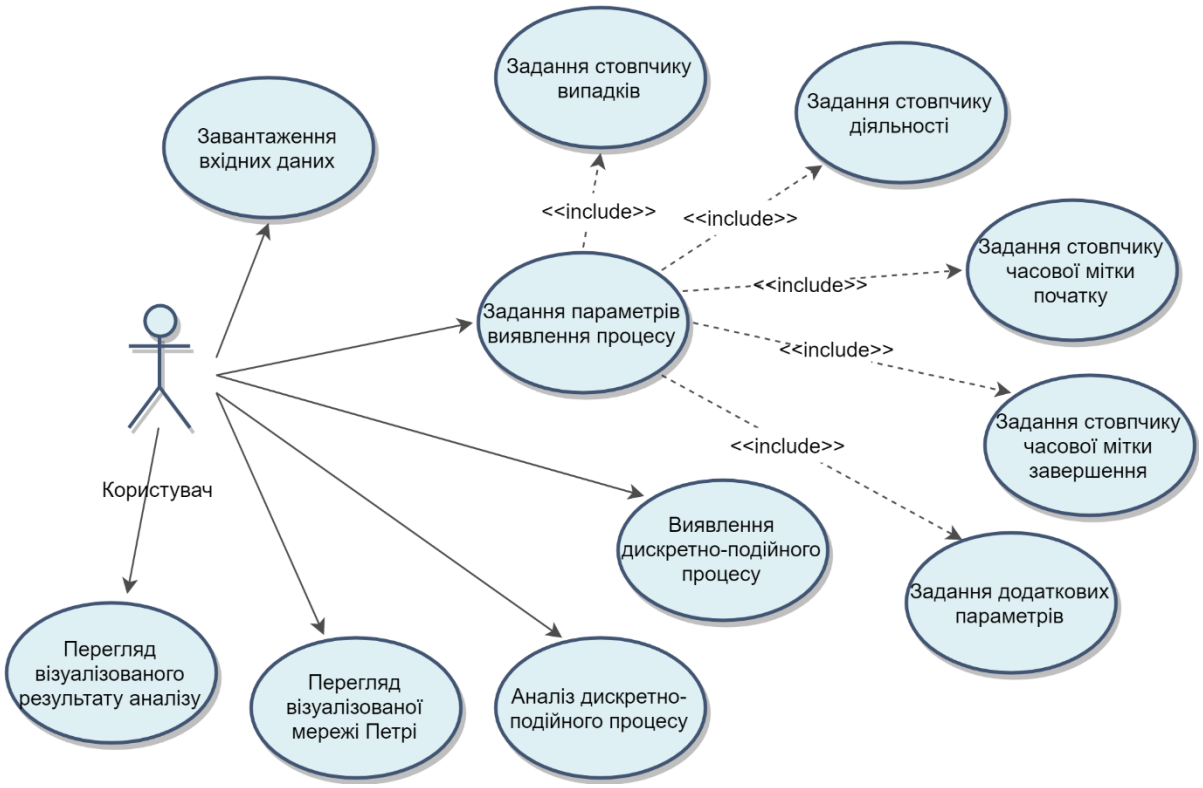


Рисунок 3.2 – Схема структурна варіантів використання

Усі варіанти використання системи задіяні одним актором – користувачем. Користувач має можливість завантажувати вхідні дані; задавати параметри виявлення процесу, що в свою чергу включать в себе задання стовпчиків діяльності, випадків, часової мітки початку, часової мітки завершення та додаткових параметрів (порогових значень кількості прямих наслідувань, міри зв'язності, значення розміру вікна для встановлення розгалужень та з'єднань); виявляти дискретно-подійний процес за заданими вхідними даними та

вказаними параметрами; аналізувати виявлену модель процесу; переглядати візуалізовану мережу Петрі та візуалізовані результати проведеного аналізу виявленої моделі дискретно-подійного процесу.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Відповідно до зазначених варіантів використання було виявлено функціональні вимоги із зазначенням їх пріоритету. Результат наведено в таблиці 4.1.

Таблиця 4.1 – Функціональні вимоги

Варіант використання	Функціональна вимога	Пріоритет
Завантаження вхідних даних	1. Система повинна надавати можливість завантажувати файл вхідних даних	Високий
	1.1. Система повинна надавати можливість завантажувати файли з розширенням .csv (англ. Comma-Separated Values)	Високий
	1.1.1. Система повинна встановлювати фільтр на відображення лише файлів з розширенням .csv підчас вибору файлу для завантаження	Низький
	1.2. Система повинна візуалізувати завантажені вхідні дані у вигляді таблиці	Середній
	1.2.1. Система повинна надавати можливість змінювати порядок стовпців у візуалізованій таблиці вхідних даних	Низький
	1.2.2. Система повинна надавати можливість відсортувати у висхідному чи низхідному порядку будь-який стовпець візуалізованої таблиці вхідних даних	Низький
Задання параметрів виявлення процесу	2. Система повинна надавати можливість задання параметрів виявлення дискретно-подійного процесу	Середній

Продовження таблиці 4.1

Задання стовпчику випадків	2.1. Система повинна надавати можливість задання стовпчику випадків	Середній
	2.1.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику діяльності	2.2. Система повинна надавати можливість задання стовпчику діяльності	Середній
	2.2.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику часової мітки початку	2.3. Система повинна надавати можливість задання стовпчику часової мітки початку	Середній
	2.3.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику часової мітки завершення	2.4. Система повинна надавати можливість задання стовпчику часової мітки завершення	Середній
	2.4.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький

Продовження таблиці 4.1

Задання додаткових параметрів	2.5. Система повинна надавати можливість задання додаткових параметрів	Середній
	2.5.1. Система повинна надавати можливість задати числові параметри: порогове значення кількості прямих наслідувань, порогове значення міри зв’язності, значення розміру вікна для встановлення розгалужень та з’єднань	Середній
	2.5.2. Система повинна надавати можливість задання додаткових параметрів за допомогою «повзунків» з обмеженим діапазоном. 2.5.3. Система повинна встановлювати діапазон допустимих значень додаткових параметрів	Низький
	2.5.4. Система повинна не надавати можливість задання таких значень додаткових параметрів, що знаходять поза діапазону допустимих значень	Низький
Виявлення дискретно-подійного процесу	3. Система повинна за вхідними даними та заданими параметрами виявляти дискретно-подійний процес у вигляді мережі Петрі	Високий
	3.1. Система повинна за допомогою евристичного алгоритму виявлення моделі процесу виявляти граф залежності процесу	Високий
	3.2. Система повинна перетворювати граф залежності процесу у мережу Петрі	Високий
	3.3. Система повинна спрощувати отриману мережу Петрі	Середній

Продовження таблиці 4.1

Аналіз дискретно-подійного процесу	4. Система повинна за вхідними даними та заданими параметрами аналізувати дискретно-подійний процес	Високий
	4.1. Система повинна розраховувати кількість спрацьовувань кожної з виявлених діяльностей у чисельному та відсотковому вимірах	Високий
	4.2. Система повинна розраховувати кількість завершених переходів між будь-якими двома з виявлених діяльностей у чисельному та відсотковому вимірах	Високий
	4.3. Система повинна розраховувати мінімальне, середнє, медіанне, та максимальне значення тривалості виконання кожної з виявлених діяльностей	Високий
	4.4. Система повинна розраховувати мінімальне, середнє, медіанне, та максимальне значення тривалості затримок між виконанням будь-яких двох з виявлених діяльностей	Високий
Перегляд візуалізованої мережі Петрі	5. Система повинна надавати можливість переглянути виявлений дискретно-подійний процес у вигляді мережі Петрі	Високий
	5.1. Система повинна візуалізовувати отриману мережу Петрі у вигляді набору сполучених між собою «місць» та «переходів», дотримуючись усіх нотацій мереж Петрі	Високий
	5.2. Система повинна надавати можливість пересувати кожен з елементів візуалізованої мережі Петрі	Низький

Продовження таблиці 4.1

	5.3. Система повинна надавати можливість редагувати кожен елемент візуалізованої мережі Петрі	Низький
	5.4. Система повинна надавати можливість видаляти елементи візуалізованої мережі Петрі	Низький
	5.5. Система повинна надавати можливість додавати нові елементи до візуалізованої мережі Петрі	Низький
Перегляд візуалізованого результату аналізу	6. Система повинна надавати можливість переглянути візуалізовані результати аналізу дискретно-подійного процесу	Високий
	6.1. Система повинна візуалізувати результати аналізу в окремому від візуалізації мережі Петрі вікні	Середній
	6.2. Система повинна згрупувати отримані результати по тематичними групам: «Частота діяльності», «Частота переходів», «Тривалість діяльності» та «Тривалість затримок»	Середній
	6.3. Система повинна відображати як повну назву діяльності, так і її буквене позначення, введене системою	Низький
	6.4. Система повинна відображати чисельні значення оцінки часу в секундах	Низький
	6.5. Система повинна надавати можливість пересувати довільним чином стовпці кожної з таблиць	Низький
	6.6. Система повинна надавати можливість сортувати у висхідному та низхідному порядках вміст кожного стовпця кожної таблиці.	Середній

4.2 Вимоги до надійності

Вимоги до надійності системи висуваються нижче:

1. Система повинна коректно реагувати на введення користувачем неправильних вхідних даних.

1.1. Система повинна відображати повідомлення про помилку при невірно введених вхідних даних.

1.2. Система повинна відображати повідомлення про помилку при невірно введених параметрах виявлення моделі процесу.

1.3. Система повинна продовжувати своє функціонування при невірно введених користувачем даних.

4.3 Вимоги до складу і параметрів технічних засобів

Для коректної роботи даного програмного продукту до технічного забезпечення мають входити:

1. комп’ютер:

1.1. з процесором з тактовою частотою не нижче 1 ГГц;

1.2. з об’ємом оперативної пам’яті не менше 1 ГБ;

2. програмне забезпечення:

2.1. операційна система Windows 7 та вище;

2.2. Java SE Runtime Environment 8 та вище;

3. комп’ютерна периферія:

3.1. монітор;

3.2. мишка.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Етапи виконання дипломної роботи подано в таблиці 5.1.

Таблиця 5.1 – Стадії та етапи розробки

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*20.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*29.02.2019*	
3.	*Постановка та формалізація задачі*	*05.03.2019*	
4.	*Розробка інформаційного забезпечення*	*15.03.2019*	
5.	*Алгоритмізація задачі*	*12.04.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*18.04.2019*	
7.	*Розробка програмного забезпечення*	*20.05.2019*	
8.	*Налагодження програми*	*27.05.2019*	
9.	*Виконання графічних документів*	*15.05.2019*	
10.	*Оформлення пояснювальної записки*	*27.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1 Види випробувань

Метою випробувань являється перевірка відповідності функцій розробленого програмного продукту встановленим функціональним та нефункціональним вимогам.

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проекту
_____ *І.В. Стеценко*
(підпис) (ініціали, прізвище)
“13” травня 2019 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“14” травня 2019 р.

Аналіз та моделювання дискретно-подійного процесу на основі журналу подій

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5123.1181-с.ПМВ*

на 18 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Розроблена система отримала наступне найменування: «Програмний продукт з аналізу та моделювання дискретно-подійного процесу на основі журналу подій».

1.2 Область застосування

Запропонований програмний продукт призначений для виявлення, аналізу та візуалізації моделі дискретно-подійного процесу на основі журналу подій.

Мета роботи – розробка, дослідження та використання реалізації евристичного алгоритму виявлення процесу (англ. Heuristics Miner) для аналізу та моделювання дискретно-подійного процесу.

Для реалізації поставленої мети роботи необхідно розв’язати наступні задачі:

- розробка та реалізація евристичного алгоритму виявлення моделі процесів;
- розробка та реалізація алгоритму перетворення виявленого процесу у мережу Петрі;
- створення засобів проведення аналізу моделі дискретно-подійного процесу на основі журналу подій;
- створення засобів графічної візуалізації виявленої моделі та результатів її аналізу.

1.3 Умовне позначення програми

Найменування «Програмний продукт з аналізу та моделювання дискретно-подійного процесу на основі журналу подій» може бути замінено на «Програмний продукт» або «ПП».

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань являється перевірка відповідності розробленого програмного продукту варіантам використання (рис. 2.1) та функціональним вимогам, представленим у технічному завданні на створення програмного продукту для аналізу та моделювання дискретно-подійного процесу на основі журналу подій.

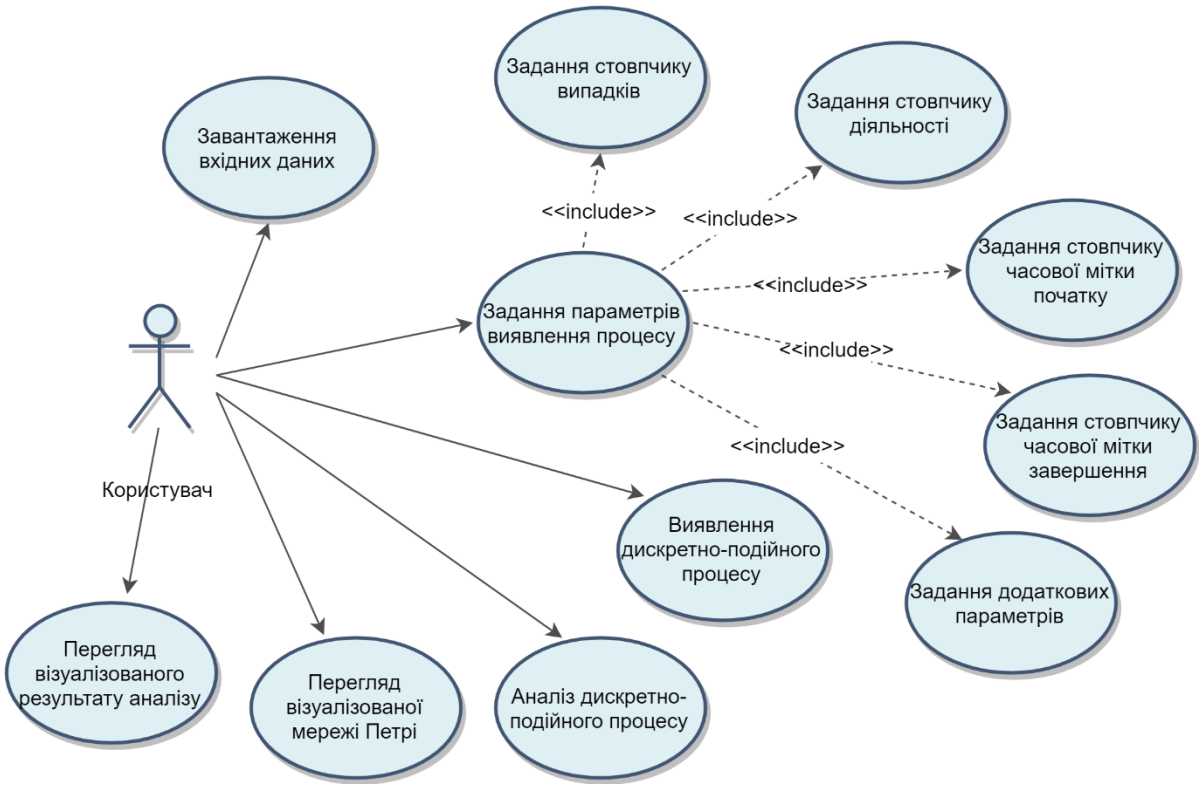


Рисунок 2.1 – Схема структурна варіантів використання

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Функціональні вимоги було сформовано відповідно до зазначених варіантів використання. Результат наведено в таблиці 3.1.

Таблиця 3.1 – Функціональні вимоги

Варіант використання	Функціональна вимога	Пріоритет
Завантаження вхідних даних	1. Система повинна надавати можливість завантажувати файл вхідних даних	Високий
	1.1. Система повинна надавати можливість завантажувати файли з розширенням .csv (англ. Comma-Separated Values)	Високий
	1.1.1. Система повинна встановлювати фільтр на відображення лише файлів з розширенням .csv підчас вибору файлу для завантаження	Низький
	1.2. Система повинна візуалізувати завантажені вхідні дані у вигляді таблиці	Середній
	1.2.1. Система повинна надавати можливість змінювати порядок стовпців у візуалізованій таблиці вхідних даних	Низький
	1.2.2. Система повинна надавати можливість відсортувати у висхідному чи низхідному порядку будь-який стовпець візуалізованої таблиці вхідних даних.	Низький
Задання параметрів виявлення процесу	2. Система повинна надавати можливість задання параметрів виявлення дискретно-подійного процесу	Середній

Продовження таблиці 3.1

Задання стовпчику випадків	2.1. Система повинна надавати можливість задання стовпчику випадків	Середній
	2.1.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику діяльності	2.2. Система повинна надавати можливість задання стовпчику діяльності	Середній
	2.2.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику часової мітки початку	2.3. Система повинна надавати можливість задання стовпчику часової мітки початку	Середній
	2.3.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький
Задання стовпчику часової мітки завершення	2.4. Система повинна надавати можливість задання стовпчику часової мітки завершення	Середній
	2.4.1. Система повинна надавати можливість обрати один з представлених стовпців візуалізованої таблиці вхідних даних за допомогою випадаючого списку з зазначеними назвами кожного стовпця	Низький

Продовження таблиці 3.1

Задання додаткових параметрів	2.5. Система повинна надавати можливість задання додаткових параметрів	Середній
	2.5.1. Система повинна надавати можливість задати числові параметри: порогове значення кількості прямих наслідувань, порогове значення міри зв'язності, значення розміру вікна для встановлення розгалужень та з'єднань	Середній
	2.5.2. Система повинна надавати можливість задання додаткових параметрів за допомогою «повзунків» з обмеженим діапазоном. 2.5.3. Система повинна встановлювати діапазон допустимих значень додаткових параметрів	Низький
	2.5.4. Система повинна не надавати можливість задання таких значень додаткових параметрів, що знаходять поза діапазону допустимих значень	Низький
Виявлення дискретно-подійного процесу	3. Система повинна за вхідними даними та заданими параметрами виявляти дискретно-подійний процес у вигляді мережі Петрі.	Високий
	3.1. Система повинна за допомогою евристичного алгоритму виявлення моделі процесу виявляти граф залежності процесу	Високий
	3.2. Система повинна перетворювати граф залежності процесу у мережу Петрі	Високий
	3.3. Система повинна спрощувати отриману мережу Петрі	Середній

Продовження таблиці 3.1

Аналіз дискретно-подійного процесу	4. Система повинна за вхідними даними та заданими параметрами аналізувати дискретно-подійний процес	Високий
	4.1. Система повинна розраховувати кількість спрацьовувань кожної з виявлених діяльностей у чисельному та відсотковому вимірах	Високий
	4.2. Система повинна розраховувати кількість завершених переходів між будь-якими двома з виявлених діяльностей у чисельному та відсотковому вимірах	Високий
	4.3. Система повинна розраховувати мінімальне, середнє, медіанне, та максимальне значення тривалості виконання кожної з виявлених діяльностей	Високий
	4.4. Система повинна розраховувати мінімальне, середнє, медіанне, та максимальне значення тривалості затримок між виконанням будь-яких двох з виявлених діяльностей	Високий
Перегляд візуалізованої мережі Петрі	5. Система повинна надавати можливість переглянути виявлений дискретно-подійний процес у вигляді мережі Петрі	Високий
	5.1. Система повинна візуалізовувати отриману мережу Петрі у вигляді набору сполучених між собою «місць» та «переходів», дотримуючись усіх нотацій мереж Петрі	Високий
	5.2. Система повинна надавати можливість пересувати кожен з елементів візуалізованої мережі Петрі	Низький

Продовження таблиці 3.1

	5.3. Система повинна надавати можливість редагувати кожен елемент візуалізованої мережі Петрі	Низький
	5.4. Система повинна надавати можливість видаляти елементи візуалізованої мережі Петрі	Низький
	5.5. Система повинна надавати можливість додавати нові елементи до візуалізованої мережі Петрі	Низький
Перегляд візуалізованого результату аналізу	6. Система повинна надавати можливість переглянути візуалізовані результати аналізу дискретно-подійного процесу	Високий
	6.1. Система повинна візуалізувати результати аналізу в окремому від візуалізації мережі Петрі вікні	Середній
	6.2. Система повинна згрупувати отримані результати по тематичними групам: «Частота діяльності», «Частота переходів», «Тривалість діяльності» та «Тривалість затримок»	Середній
	6.3. Система повинна відображати як повну назву діяльності, так і її буквене позначення, введене системою	Низький
	6.4. Система повинна відображати чисельні значення оцінки часу в секундах	Низький
	6.5. Система повинна надавати можливість пересувати довільним чином стовпці кожної з таблиць	Низький
	6.6. Система повинна надавати можливість сортувати у висхідному та низхідному порядках вміст кожного стовпця кожної таблиці	Середній

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Висуваються наступні вимоги до програмної документації:

- програмна документація повинна містити пояснювальну записку, що включає в себе такі розділи, як «Загальні положення», «Інформаційне забезпечення», «Математичне забезпечення», «Програмне та технічне забезпечення»;
- програмна документація повинна містити детальне керівництво користувача з покроковим описом та знімками екранних форм виконання усіх зазнаних в технічному завданні варіантів використання програмного продукту;
- програмна документація повинна містити структурну діаграму класів розробленого програмного продукту з детальним описом їх ролі та специфікацією їх функцій.
- програмна документація повинна бути виконана з дотриманням усіх встановлених норм ведення програмної документації, включаючи та не обмежуючись документом «ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів».

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Перевірка працездатності розробленого програмного продукту має бути здійснена за допомогою функціонального тестування методом чорної скриньки. Порядок випробувань програмного продукту та відповідні їм функції наведено в таблиці 5.1.

Таблиця 5.1 – Порядок випробувань

№	Випробування	Функції
1	Відкриття вікна завантаження вхідних даних	Завантаження вхідних даних
2	Завантаження вхідних даних	Завантаження вхідних даних
3	Вибір стовпця ідентифікатора випадків	Задання параметрів виявлення процесу, Задання стовпчику випадків
4	Вибір стовпця мітки часу початку	Задання параметрів виявлення процесу, Задання стовпчику мітки часу початку
5	Вибір стовпця мітки часу завершення	Задання параметрів виявлення процесу, Задання стовпчику мітки часу завершення
6	Вибір стовпця міток часу початку та завершення	Задання параметрів виявлення процесу, Задання стовпчику мітки часу початку, Задання стовпчику мітки часу завершення
7	Задання параметрів виявлення процесу	Задання параметрів виявлення процесу
8	Виявлення дискретно подійного-процесу (максимальна кількість наслідувань)	Виявлення дискретно подійного-процесу
9	Виявлення дискретно подійного-процесу (максимальна міра зв’язності)	Виявлення дискретно подійного-процесу, Перегляд візуалізованої мережі Петрі
10	Виявлення дискретно подійного-процесу (значення параметрів за замовченням)	Виявлення дискретно подійного-процесу, Перегляд візуалізованої мережі Петрі

Продовження таблиці 5.1

11	Аналіз дискретно подійного-процесу	Аналіз дискретно подійного-процесу,
12	Перегляд результатів аналізу	Перегляд візуалізованого результатату аналізу

6 МЕТОДИ ВИПРОБУВАНЬ

6.1 Функціональне тестування

У таблицях 6.1-6.12 наведено перелік, опис та хід випробувань основних функціональних можливостей розробленого програмного продукту.

Таблиця 6.1 – Відкриття вікна завантаження вхідних даних

Мета тесту:	**Перевірка функції «Завантаження вхідних даних»**
Початковий стан ПП	Відкрите головне вікно ПП
Вхідні дані:	Файл журналу подій розширення *.csv
Схема проведення тесту:	Обрати підпункт Open Event Data пункту Process Mining з головного меню.
Очікуваний результат:	Відкрите вікно вибору файлу Open a file
Стан ПП після проведення випробувань:	Відкрите вікно вибору файлу Open a file

Таблиця 6.2 – Завантаження вхідних даних

Мета тесту:	**Перевірка функції «Завантаження вхідних даних»**
Початковий стан ПП	Відкрите вікно вибору файлу Open a file
Вхідні дані:	Файл журналу подій з розширенням *.csv
Схема проведення тесту:	Обрати файл журналу подій з розширенням *.csv для завантаження. Натиснути кнопку «Відкрити»
Очікуваний результат:	Відкрите вікно перегляду журналу подій
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 6.3 – Вибір стовпця ідентифікатора випадків

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні данні:	Стовпець з назвою activity title

Продовження таблиці 6.3

Схема проведення тесту:	Визначити параметр case id column як activity title. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обраний стовпець ідентифікатора подій
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 6.4 – Вибір стовпця мітки часу початку

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою activity title
Схема проведення тесту:	Визначити параметр start timestamp column як activity title. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обраний стовпець мітки часу початку
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 6.5 – Вибір стовпця мітки часу завершення

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою activity title
Схема проведення тесту:	Визначити параметр finish timestamp column як activity title. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обраний стовпець мітки часу завершення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 6.6 – Вибір стовпця міток часу початку та завершення

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою start time, finish time
Схема проведення тесту:	Визначити параметр start timestamp column як finish time, а параметр finish timestamp column як start time. Натиснути кнопку Next Step
Очікуваний результат:	Повідомлення про некоректно обрані стопці міток часу початку та завершення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій

Таблиця 6.7 – Задання параметрів виявлення процесу

Мета тесту:	**Перевірка функції «Задання параметрів виявлення процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій
Вхідні дані:	Стовпець з назвою case id, activity title, start time, finish time
Схема проведення тесту:	Визначити параметр case id column як case id; параметр activity column як activity title, start timestamp column як start time, а параметр finish timestamp column як finish time. Натиснути кнопку Next Step
Очікуваний результат:	Перехід до наступного кроку виявлення. Зображення таблиць назв діяльності та її скорочення; послідовність подій та їх частота, нові параметри для введення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій на другому кроці

Таблиця 6.8 – Виявлення дискретно подійного-процесу 1

Мета тесту:	**Перевірка функції «Виявлення дискретно подійного-процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій на другому кроці
Вхідні дані:	Значення параметру succession threshold = 197 (максимум повзунку)
Схема проведення тесту:	Визначити параметр succession threshold = 197. Натиснути кнопку Generate Petri net
Очікуваний результат:	Повідомлення про помилку виявлення процесу, так як виявлена мережа незв’язна. Рекомендація змінити параметри виявлення
Стан ПП після проведення випробувань:	Відкрите вікно перегляду журналу подій на другому кроці

Таблиця 6.9 – Виявлення дискретно подійного-процесу 2

Мета тесту:	**Перевірка функції «Виявлення дискретно подійного-процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій на другому кроці
Вхідні дані:	Значення параметру succession threshold = 1, dependency threshold = 1.00
Схема проведення тесту:	Визначити параметр succession threshold = 1, dependency threshold = 1.00. Натиснути кнопку Generate Petri net
Очікуваний результат:	Візуалізація у головному вікні програмного продукту мережі Петрі, що складається з двох місць та одного переходу, послідовно з’єднаних між собою
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі та вікно перегляду журналу подій

Таблиця 6.10 – Виявлення дискретно подійного-процесу 3

Мета тесту:	**Перевірка функції «Виявлення дискретно подійного-процесу»**
Початковий стан ПП	Відкрите вікно перегляду журналу подій на другому кроці
Вхідні дані:	Значення параметру succession threshold = 1, dependency threshold = 0.5, window size = 4.
Схема проведення тесту:	Визначити параметр succession threshold = 1, dependency threshold = 0.5, window size = 4. Натиснути кнопку Generate Petri net
Очікуваний результат:	Візуалізація у головному вікні програмного продукту мережі Петрі
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі та вікно перегляду журналу подій

Таблиця 6.11 – Аналіз дискретно подійного-процесу

Мета тесту:	**Перевірка функції «Аналіз дискретно подійного-процесу»**
Початковий стан ПП	Відкрите головне вікно з мережею Петрі та вікно перегляду журналу подій.
Вхідні дані:	
Схема проведення тесту:	Натиснути кнопку Generate Report
Очікуваний результат:	Відкриття вікна звіту з аналізу дискретно-подійного процесу
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі, вікно перегляду журналу подій та вікно звіту

Таблиця 6.12 – Перегляд результатів аналізу

Мета тесту:	**Перевірка функції «Перегляд результатів аналізу»**
Початковий стан ПП	Відкрите головне вікно з мережею Петрі, вікно перегляду журналу подій та вікно звіту.
Вхідні дані:	

Продовження таблиці 6.12

Схема проведення тесту:	У вікні звіту з аналізу почергово відкрити кожну з чотирьох представлених вкладок (розділів) звіту
Очікуваний результат:	Кожна вкладка (розділ) звіту містить заповнену таблицю з результатами аналізу
Стан ПП після проведення випробувань:	Відкрите головне вікно з мережею Петрі, вікно перегляду журналу подій та вікно звіту

Графічний матеріал
до дипломного проекту

на тему: Аналіз та моделювання дискретно-подійного процесу на основі журналу подій

Київ – 2019 року

Рішення з математичного забезпечення

Теоретичні основи методу виявлення процесу

Нехай E – набір подій, E* – набір всіх послідовностей, які складаються з нуля або більше подій з E. Тоді $\sigma \in E^*$ це довільна послідовність подій.

Нехай W буде журналом подій над E, тобто $W \subseteq E^*$.

Нехай $a, b \in E$, тоді:

1. $a >_W b$ якщо існує послідовність $\sigma = e_1 e_2 e_3 \ldots e_n$ та $i \in \{1, \ldots, n-1\}$ такий, що $\sigma \in W$ та $e_i = a$ та $e_i+1 = b$,

2. $a \rightarrow_W b$ якщо $a >_W b$ та не $b >_W a$,

3. $a \#_W b$ якщо не $a >_W$ b та не $b >_W a$,

4. $a \|_W b$ якщо $a >_W b$ та $b >_W a$,

5. $a >>_W b$ якщо існує послідовність $\sigma = e_1 e_2 e_3 \ldots e_n$ та $i \in \{1, \ldots, n-2\}$ такий, що $\sigma \in W$ та $e_i = a$ та e_i+1 = b та $e_i + 2 = a$,

6. $a>>>_W b$, якщо існує послідовність $\sigma = e_1 e_2 e_3 \ldots e_n$ такий, що $\sigma \in W$ та $e_i = a$ та $e_j = b$.

Нехай $a \Rightarrow_W b$ – показник, заснований на частоті подій, що використовується для позначення того, наскільки ми певні, що існує дійсно залежність між двома подіями A та B.

$$a \Rightarrow_w b = \frac{|a>_w b| - |b>_w a|}{|a>_w b| + |b>_w a| + 1}$$

$$a \Rightarrow_w a = \frac{|a>_w a|}{|a>_w a| + 1}$$

Мережа Петрі з трійкою $N = (P, T, F)$, де P – скінченна множина *позицій*, T - скінчена множина *переходів*, таких, що $P \cap T = \emptyset$, і $F \subseteq (P \times T) \cup (T \times P)$ - сукупність спрямованих дуг.

Задача полягає в знаходженні такої мережі Петрі N = (P, T, F), для якої виконується:

$$E_n \subseteq T,$$

де

$$E_n = \{a, b \in E \mid (a >_W b) \geq s \wedge (a \Rightarrow_W b) \geq d\},$$

s – порогове значення прямих слідувань,

d – порогове значення міри зв’язності.

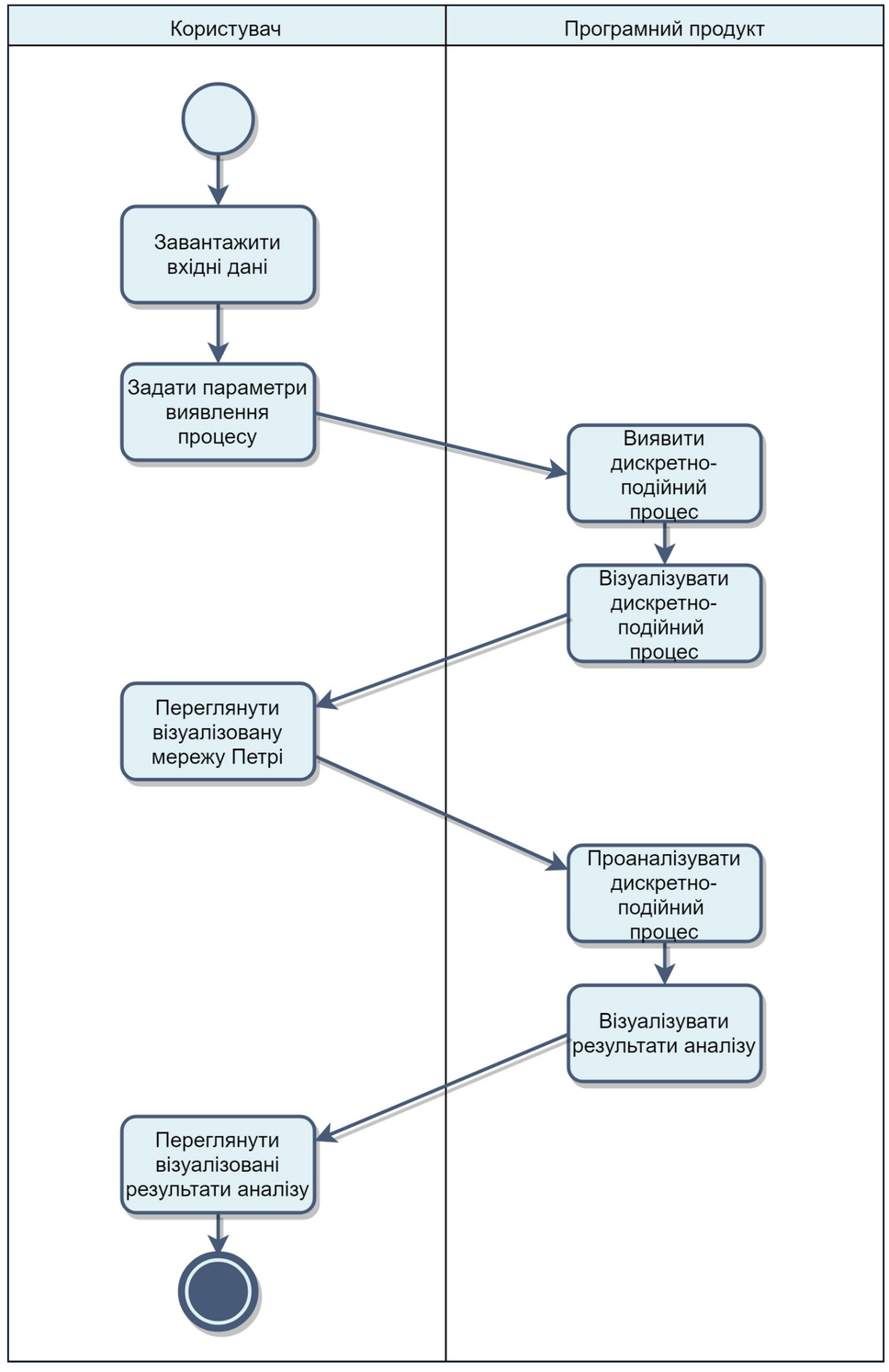
Демонстраційний плакат до дипломного проекту

„Аналіз та моделювання дискретно-подійного процесу на основі журналу подій”

Виконав студент гр. ІС-51 Рябцев С.В.

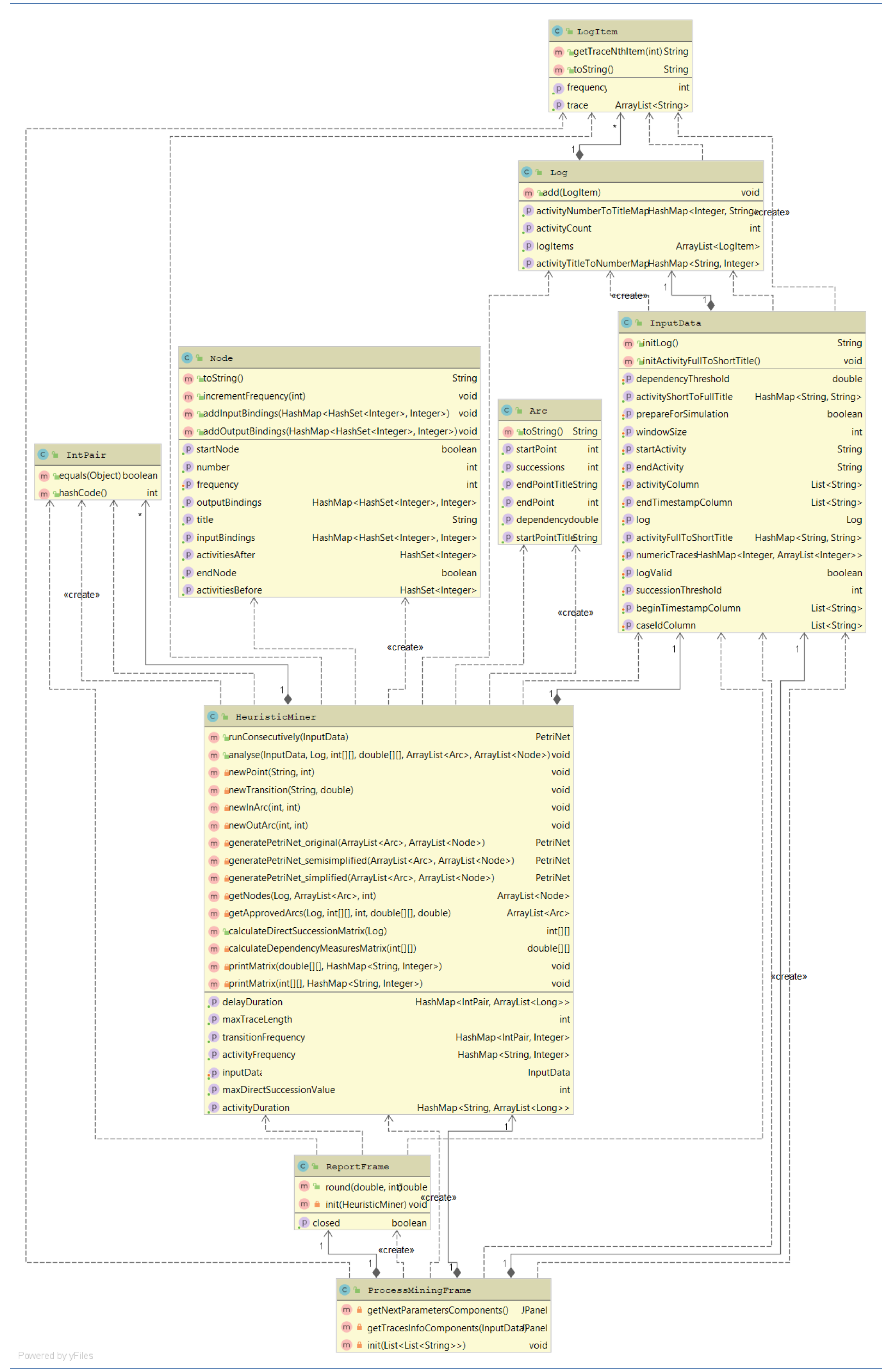
Керівник ДП Стеценко І.В.

ДП ІС-5123.1181-с.ССД



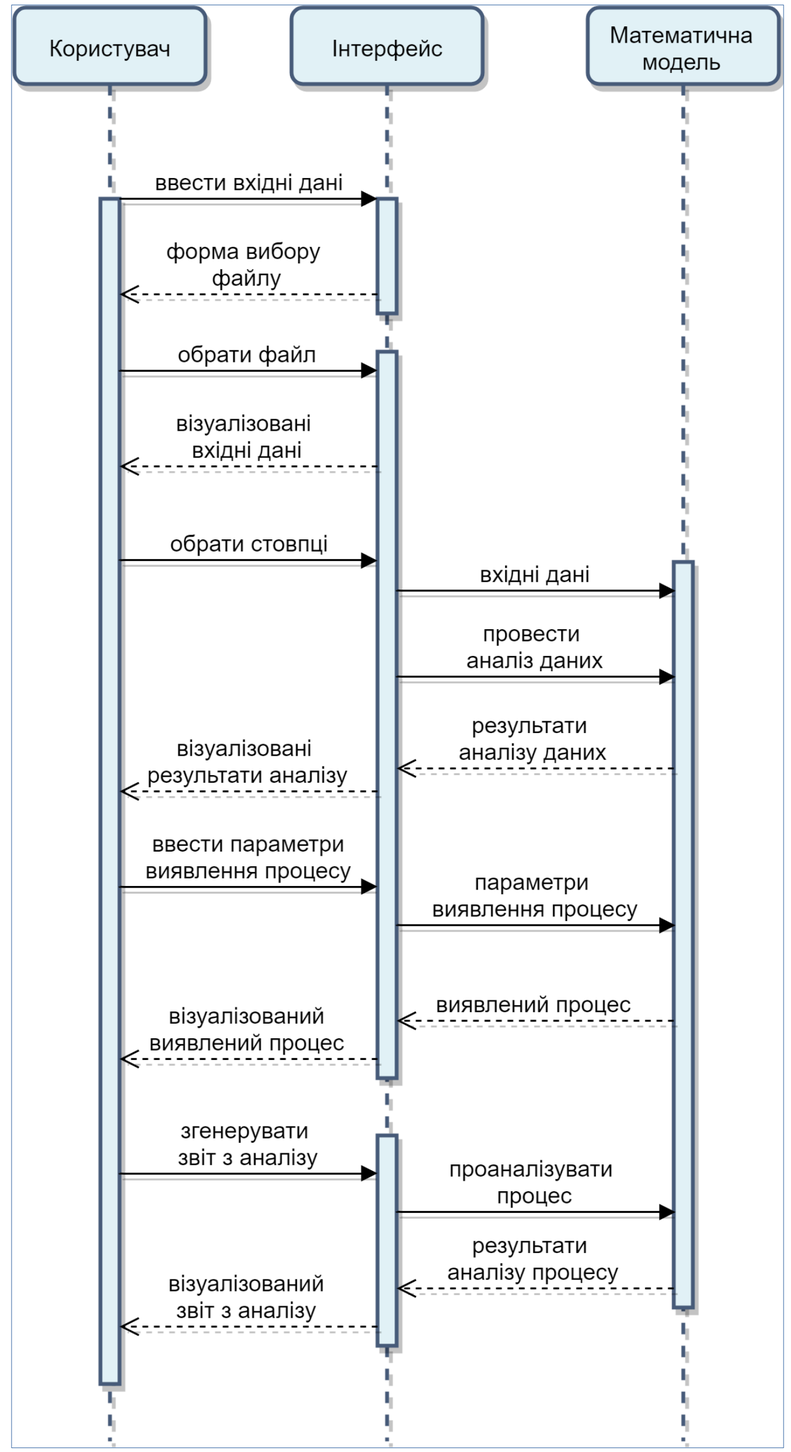
Зм.	Арк.	№ докум.	Підп.	Дата	ДП ІС-5123.1181-с.ССД		
Розроб.		Рябцев С.В.			Схема структурна діяльності	Лит.	Маса	Масштаб
Перев.		Стеценко І.В.						
Т. Кон.						Аркуш 1	Аркушів 1	
Н. Кон.		Москаленко Н.В.			Аналіз та моделювання дискретно-подійного процесу на основі журналу подій	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51		
Затв.		Стеценко І.В.						

ДП ІС-5123.1181-с.ССК



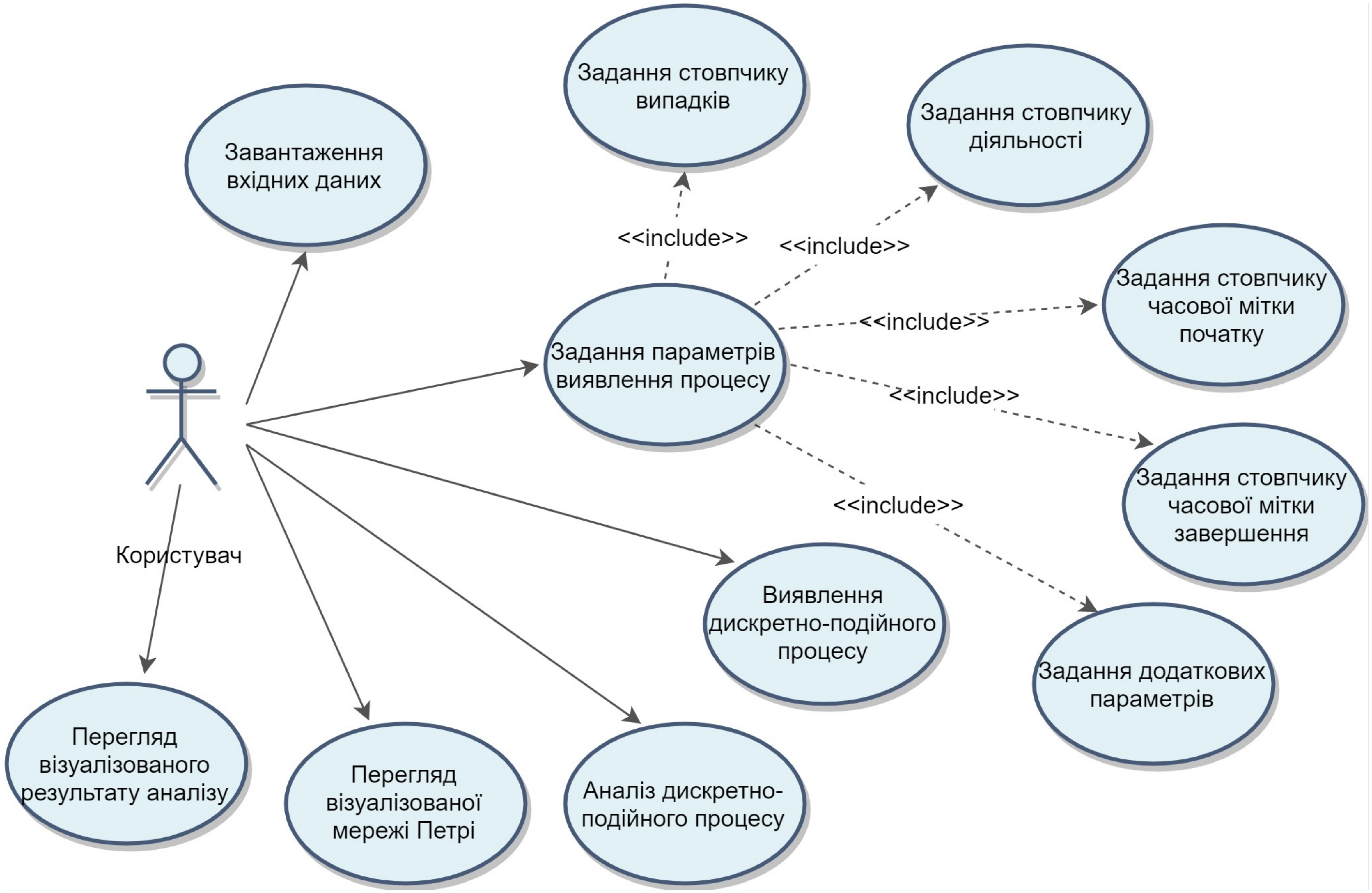
					ДП ІС-5123.1181-с.ССК		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Рябцев С.В.			Схема структурна класів програмного забезпечення	Лит.	Маса / Масштаб
Перев.		Стеценко І.В.					
Т. Кон.						Аркуш 1	Аркушів 1
Н. Кон.		Москаленко Н.В.			Аналіз та моделювання дискретно-подійного процесу на основі журналу подій	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Затв.		Стеценко І.В.					

ДП ІС-5123.1181-с.ССП



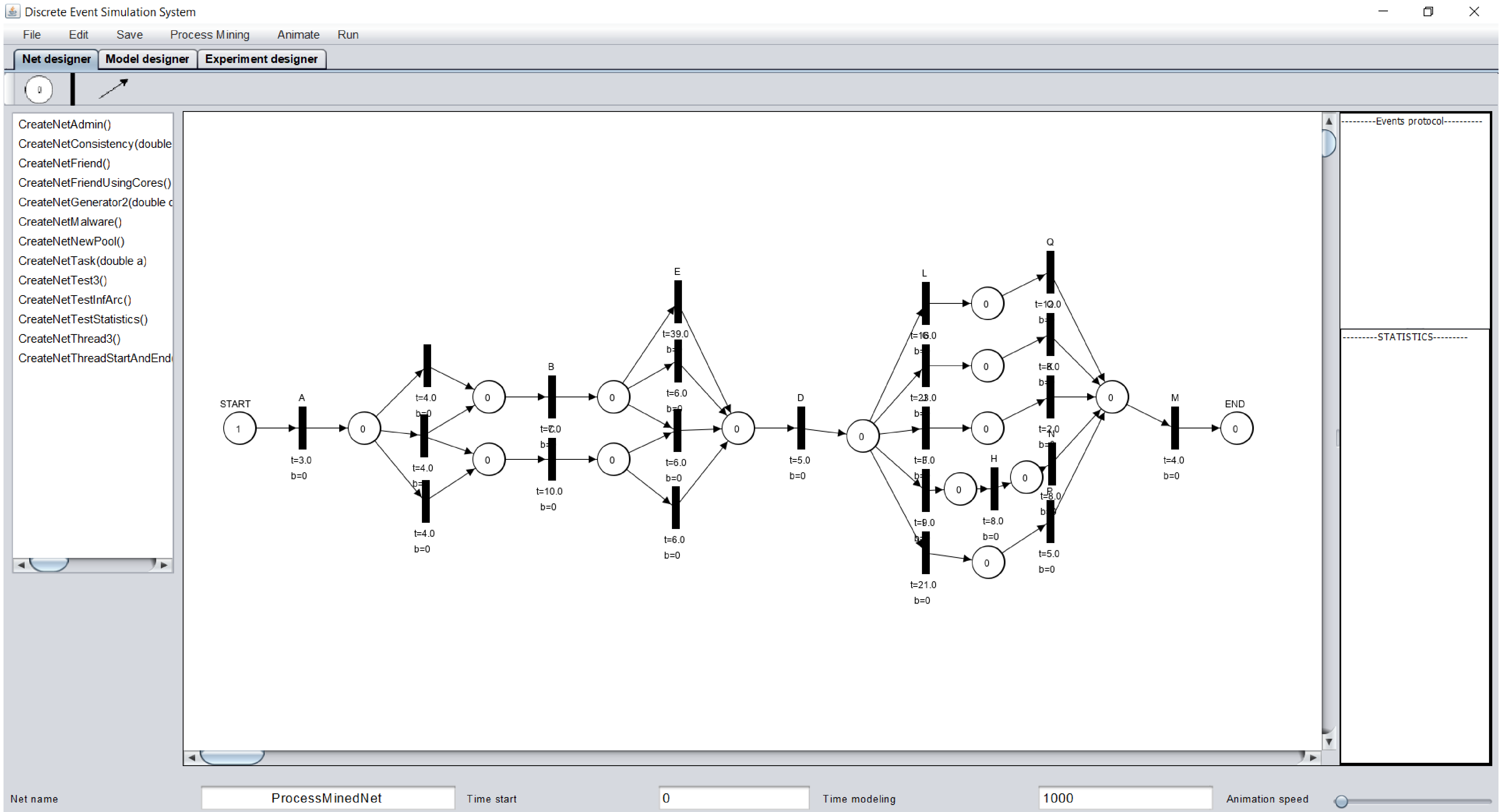
Зм.	Арк.	№ докум.	Підп.	Дата	ДП ІС-5123.1181-с.ССП		
Розроб.		Рябцев С.В.			Схема структурна послідовності	Лит.	Маса	Масштаб
Перев.		Стеценко І.В.						
Т. Кон.						Аркуш 1	Аркушів 1	
Н. Кон.		Москаленко Н.В.			Аналіз та моделювання дискретно-подійного процесу на основі журналу подій	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51		
Затв.		Стеценко І.В.						

ДП ІС-5123.1181-с.ССВ



					ДП ІС-5123.1181-с.ССВ			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Рябцев С.В.			*Схема структурна варіантів використання*	Літера	Маса	Масштаб
Перевірив		Стеценко І.В.						
Т. кон.						Аркуш 1		Аркушів 1
					Аналіз та моделювання дискретно-подійного процесу на основі журналу подій	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*		
Н. кон.		Москаленко Н.В.						
Затвердив		Стеценко І.В.						

ДП ІС-5123.1181-с.КЕ



Зм.	Арк.	№ документа	Підпис	Дата	ДП ІС-5123.1181-с.КЕ		
Розробив		Рябцев С.В.			*Креслення вигляду екранних форм*	Літера	Маса	Масштаб
Перевірив		Стеценко І.В.				Аркуш 1	Аркушів 1
Т. кон.							
					Аналіз та моделювання дискретно-подійного процесу на основі журналу подій	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*	
Н. кон.		Москаленко Н.В.					
Затвердив		Стеценко І.В.					

Analysis Report

Activity Frequency | Transition Frequency | Activity Duration | Delay Duration

From, Full...	From, Sh...	To, Full Ti...	To, Short ...	Min (s)	Avg (s) ▼	Median (s)	Max (s)
Activity 9.2	O	Activity 1...	M	4.0	22.75	23.5	42.0
Activity 5.2	P	Activity 1...	M	4.0	20.92	20.0	37.0
Activity 6.2	Q	Activity 1...	M	5.0	17.67	17.0	30.0
Activity 4.1	D	Activity 6.1	L	1.0	12.98	9.0	67.0
Activity 8.2	H	Activity 8.3	N	3.0	11.16	12.0	17.0
Activity 4.1	D	Activity 7.1	J	2.0	9.35	8.5	42.0
Activity 4.1	D	Activity 8.1	F	2.0	9.11	8.0	30.0
Activity 4.1	D	Activity 5.1	I	2.0	9.1	7.0	66.0
Activity 4.1	D	Activity 9.1	G	1.0	8.48	9.0	20.0
Activity 8.3	N	Activity 1...	M	2.0	7.95	8.0	14.0
Activity 2.1	B	Activity 4.1	D	0.0	5.77	4.0	18.0
Activity 3.1	C	Activity 4.1	D	1.0	5.69	5.0	13.0
Activity 2.2	E	Activity 4.1	D	1.0	5.69	7.0	11.0
Activity 6.1	L	Activity 6.2	Q	1.0	5.23	6.0	9.0
Activity 8.1	F	Activity 8.2	H	1.0	4.89	5.0	9.0
Activity 5.1	I	Activity 5.2	P	1.0	4.75	5.0	9.0
Activity 9.1	G	Activity 9.2	O	1.0	4.43	4.0	9.0
Activity 7.1	J	Activity 7.2	K	1.0	4.38	4.0	9.0
Activity 1.1	A	Activity 2.1	B	0.0	4.05	4.0	9.0
Activity 1.1	A	Activity 3.1	C	1.0	3.84	4.0	7.0
Activity 2.1	B	Activity 3.1	C	0.0	3.18	3.0	5.0
Activity 3.1	C	Activity 2.1	B	0.0	2.94	3.0	6.0

					ДП ІС-5123.1181-с.КЗ
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду звітних форм
Розробив		Рябцев С.В.			
Перевірив		Стеценко І.В.			
Т. кон.					
Н. кон.		Москаленко Н.В.			Аналіз та моделювання дискретно-подійного процесу на основі журналу подій
Затвердив		Стеценко І.В.			

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51